



RugFreeCoins Audit



Minionaire INU Token Smart Contract Security Audit

July 23rd, 2023

Overview

- ✗ Mint function found, the owner can mint tokens after initial deployment.
- ✓ The owner can't pause trading once it's enabled
- ✓ 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 claim any BEP20 tokens including the Minionaire Inu token)
- ✗ The owner can't change fees by more than 20%. (The owner can change all fees without any limitation)
- ✗ The owner can't set a max transaction limit. (The owner can change the max tx amount to a very low value)

- **HIGH SEVERITY ISSUES**

Swapping tokens to BNB multiple times within the same function can be inefficient in terms of gas usage. To address this issue and improve gas efficiency, it is advisable to modify the swapAndLiquify function to swap all the tokens for BNB at once

```
// swap tokens for ETH
swapTokensForEth(half); // <- this breaks the ETH -> HATE swap when swap+liquify is triggered

// how much ETH did we just swap into?
uint256 newBalance = address(this).balance.sub(initialBalance);

// add liquidity to uniswap
addLiquidity(otherHalf, newBalance);

emit SwapAndLiquify(half, newBalance, otherHalf);

uint256 contractBalance = address(this).balance;
swapTokensForEth(tokensForMarketing);
uint256 transferredMBalance = address(this).balance.sub(
    contractBalance
);

uint256 contractBalanceB = address(this).balance;
swapTokensForEth(tokensForDev);
uint256 transferredBBalance = address(this).balance.sub(
    contractBalanceB
);

//Send to Marketing address
```

The owner can change all fees without any limitation, so the owner can stop trading by putting higher amounts of fees

```
function setTaxFeePercent(
    uint256 taxFee,
    uint256 liquidityFee,
    uint256 MarketingFee,
    uint256 burnFee,
    uint devFee
) external onlyOwner {
    _taxFee = taxFee;
    _pretaxfee = _taxFee;
    _liquidityFee = liquidityFee;
    _preliquidityfee = _liquidityFee;
    _marketingFee = MarketingFee;
    _premarketingfee = _marketingFee;
    _devFee = devFee;
    _predevfee = devFee;
    _burnFee = burnFee;
    _preburnfee = _burnFee;
}

function setsellTaxFeePercent(
    uint256 taxesellFee,
    uint256 liquiditysellFee,
    uint256 MarketingsellFee,
    uint256 burnsellFee,
    uint256 devsellFee
) external onlyOwner {
    _taxsellFee = taxesellFee;
    _liquiditysellFee = liquiditysellFee;
    _marketingsellFee = MarketingsellFee;
    _devsellFee = devsellFee;
    _burnsellFee = burnsellFee;
}
```

The owner can change the swapping token amount without minimum limitation, the owner can stop selling by putting this to 0.

```
function setNumTokensSellToAddToLiquidity(
    uint256 newAmt
) external onlyOwner {
    numTokensSellToAddToLiquidity = newAmt * 10 ** _decimals;
}
```

The owner can change the max tx amount to a very low value, the owner can stop selling by putting this to a very low value

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    require(maxTxAmount > 0, "Cannot set transaction amount as zero");
    _maxTxAmount = maxTxAmount * 10 ** _decimals;
}
```

The owner can change the router address and its validation the router properly, the owner can stop selling by putting the wrong router address here.

```
//No problem, just change it here!
function setRouterAddress(address newRouter) public onlyOwner {
    IUniswapV2Router02 _newPancakeRouter = IUniswapV2Router02(newRouter);
    uniswapV2Pair = IUniswapV2Factory(_newPancakeRouter.factory())
        .createPair(address(this), _newPancakeRouter.WETH());
    uniswapV2Router = _newPancakeRouter;
}
```

The owner can withdraw native tokens from the contract, the owner can stop swapping by withdrawing all native tokens.

```
function withdrawToken() external onlyOwner {
    IERC20 erc20token = IERC20(address(this));
    uint256 balance = erc20token.balanceOf(address(this));
    erc20token.transfer(owner(), balance);
}
```

The owner can mint a new token and increase the total supply, but those tokens will not go to owner's wallet.

```
function _bburn(address account, uint256 amount) internal virtual {
    require(account != address(0), "BEP20: mint to the zero address");

    _beforeTokenTransfer(address(0), account, amount);

    _tTotal += amount;
    emit Transfer(address(0), account, amount);
}

function Burn(uint256 amount) public onlyOwner {
    _bburn(msg.sender, amount);
}
```

Contents

Overview	ii
Audit details	1
Disclaimer	2
Background	3
Roadmap	4
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	19
Owner privileges	25
Audit conclusion	25

Audit details



Audited project
Minionaire Inu Token



Contract Address
0x5daC92BE671dAbc5112EbE5e5c5C5e1BaB116296



Client contact
Minionaire Inu Token Team



Blockchain
Binance Smart chain



Project website
<https://minionaireinu.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 Minionaire Inu Token Team to perform an audit of the smart contract.

<https://bscscan.com/token/0x5dac92be671dabc5112ebe5e5c5c5e1bab116296>

This audit focuses on verifying 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.

Roadmap

Phase 01

- Conducting full market analysis and research.
- Project implementation and Whitepaper creation.
- Website creation and social media setup.
- Token creation and Smart Contract development.
- Launching a massive marketing campaign targeting worldwide investors.
- Develop strategies to ensure long-term sustainability and stability.

Phase 02

- Develop partnerships to increase the visibility and adoption of the token.
- Listing on multiple launch pads.
- Development of our Staking algorithms.

Phase 03

- Development of our NFT marketplace.
- Implementation of burning mechanism.
- Designing and developing the play-to-earn game.

Phase 04

- Integration of gaming application with NFT marketplace.
- Launching the game on the three major mobile operating systems.
- Listing on additional exchanges.

Phase 05

- Listing on mainstream exchanges.
- Developing additional partnerships.
- Exploration of additional innovative ideas supporting the project's sustainability.

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 interested in taking part in the Minionaire Inu token ecosystem.
- Anyone who's interested in taking part in the future plans of Minionaire Inu Token.
- Anyone who's interested in making financial transactions with any other party using Minionaire Inu Token as the currency.

Potential to grow with score points

1.	Project efficiency	8/10
2.	Project uniqueness	8/10
3	Information quality	8/10
4	Service quality	8/10
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	5/10
10	Smart contract functionality assessment	8/10
Total Points		7.8/10

Contract details

Token contract details for 23rd of July 2023

Contract name	Minionaire Inu
Contract address	0x5daC92BE671dAbc5112EbE5e5c5C5e1BaB116296
Token supply	100,000,000,000,000
Token ticker	MNUR
Decimals	18
Token holders	1
Transaction count	2
Contract deployer address	0x51D62ce79cEf276B3210e040409229ae864b8A70
Contract 's current owner address	0x45a64b6abca89894c3755a11dbed841b942dd6e3
Marketing wallet	0xbeda3cf55117c1294f3ed919e0bbcb3b7fb5f323
Dev wallet	0xd97f869113cefba4f1864bde92b4a1b270eb1610











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	High 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 & centralization	Access control of owners	High Issue
		Business logics	Medium Issue
		Business implementations	Medium Issue
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
IERC20	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 !
SafeMath	Library			
L	add	Internal 		
L	sub	Internal 		
L	sub	Internal 		
L	mul	Internal 		
L	div	Internal 		
L	div	Internal 		
L	mod	Internal 		










L	mod	Internal 🔒		
Context	Implementation			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
Address	Library			
L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	●	
L	functionCall	Internal 🔒	●	
L	functionCall	Internal 🔒	●	
L	functionCallWithValue	Internal 🔒	●	
L	functionCallWithValue	Internal 🔒	●	
L	_functionCallWithValue	Private 🔒	●	
Ownable	Implementation	Context		
L		Public !	●	NO !
L	owner	Public !		NO !
L	renounceOwnership	Public !	●	onlyOwner
L	transferOwnership	Public !	●	onlyOwner
L	geUnlockTime	Public !		NO !
L	lock	Public !	●	onlyOwner

L	unlock	Public !		NO !
IUniswapV2 Factory	Interface			
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 !
IUniswapV2 Pair	Interface			
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	burn	External !		NO !
L	swap	External !		NO !
L	skim	External !		NO !
L	sync	External !		NO !
L	initialize	External !		NO !
IUniswapV2 Router01	Interface			
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 !
IUniswapV2Router02	Interface	IUniswapV2Router01		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External !		NO !

L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External !		NO !
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !		NO !
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External !		NO !
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !		NO !
Minionaire Inu	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	setMaxWalletToken	External !		onlyOwner
L	_bburn	Internal 		
L	Burn	Public !		onlyOwner
L	_beforeTokenTransfer	Internal 		
L	approve	Public !		NO !
L	transferFrom	Public !		NO !

L	increaseAllowance	Public !		NO !
L	decreaseAllowance	Public !		NO !
L	isExcludedFromReward	Public !		NO !
L	totalFees	Public !		NO !
L	deliver	Public !		NO !
L	reflectionFromToken	Public !		NO !
L	tokenFromReflection	Public !		NO !
L	excludeFromReward	Public !		onlyOwner
L	includeInReward	External !		onlyOwner
L	_transferBothExcluded	Private 		
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	calculateTaxFee	Private 		
L	calculateLiquidityFee	Private 		
L	removeAllFee	Private 		

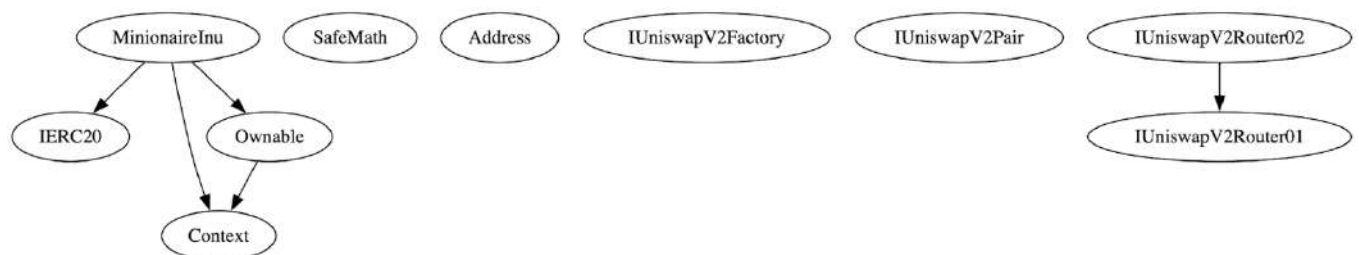
L	gg	Private 🗝️	🔴	
L	ggbb	Private 🗝️	🔴	
L	restoreAllFee	Private 🗝️	🔴	
L	isExcludedFromFee	Public !		NO !
L	_approve	Private 🗝️	🔴	
L	_transfer	Private 🗝️	🔴	
L	swapAndLiquify	Private 🗝️	🔴	lockThe Swap
L	swapTokensForEth	Private 🗝️	🔴	
L	addLiquidity	Private 🗝️	🔴	
L	_tokenTransfer	Private 🗝️	🔴	
L	_transferStandard	Private 🗝️	🔴	
L	takeBurn	Private 🗝️	🔴	
L	takeMarketing	Private 🗝️	🔴	
L	takedev	Private 🗝️	🔴	
L	_transferToExcluded	Private 🗝️	🔴	
L	_transferFromExcluded	Private 🗝️	🔴	
L	afterPresale	External !	🔴	onlyOwner
L	excludeFromFee	Public !	🔴	onlyOwner
L	includeInFee	Public !	🔴	onlyOwner
L	setMarketingWallet	External !	🔴	onlyOwner
L	setdevWallet	External !	🔴	onlyOwner

L	setTaxFeePercent	External !	●	onlyOwner
L	setsellTaxFeePercent	External !	●	onlyOwner
L	setNumTokensSellToAddToLiquidity	External !	●	onlyOwner
L	setMaxTxAmount	External !	●	onlyOwner
L	setRouterAddress	Public !	●	onlyOwner
L	setSwapAndLiquifyEnabled	Public !	●	onlyOwner
L	transferToAddressETH	Private 🗝️	●	
L	tokenSale	Public !	💵	NO !
L	setSalePrice	External !	●	onlyOwner
L	claim	Public !	●	onlyOwner
L	withdrawToken	External !	●	onlyOwner

Legend

Symbol	Meaning
●	Function can modify state
💵	Function is payable

Inheritance Hierarchy



Security issue checking status

❖ High severity issues

Swapping tokens to BNB multiple times within the same function can be inefficient in terms of gas usage. To address this issue and improve gas efficiency, it is advisable to modify the swapAndLiquify function to swap all the tokens for BNB at once

```
// swap tokens for ETH
swapTokensForEth(half); // <- this breaks the ETH -> HATE swap when swap+liquify is triggered

// how much ETH did we just swap into?
uint256 newBalance = address(this).balance.sub(initialBalance);

// add liquidity to uniswap
addLiquidity(otherHalf, newBalance);

emit SwapAndLiquify(half, newBalance, otherHalf);

uint256 contractBalance = address(this).balance;
swapTokensForEth(tokensForMarketing);
uint256 transferredMBalance = address(this).balance.sub(
    contractBalance
);

uint256 contractBalanceB = address(this).balance;
swapTokensForEth(tokensFordev);
uint256 transferredBBalance = address(this).balance.sub(
    contractBalanceB
);

//Send to Marketing address
```

The owner can change all fees without any limitation, so the owner can stop trading by putting higher amounts of fees

```
function setTaxFeePercent(
    uint256 taxFee,
    uint256 liquidityFee,
    uint256 MarketingFee,
    uint256 burnFee,
    uint devFee
) external onlyOwner {
    _taxFee = taxFee;
    _pretaxfee = _taxFee;
    _liquidityFee = liquidityFee;
    _preliquidityfee = _liquidityFee;
    _marketingFee = MarketingFee;
    _premarketingfee = _marketingFee;
    _devFee = devFee;
    _predevfee = devFee;
    _burnFee = burnFee;
    _preburnfee = _burnFee;
}

function setsellTaxFeePercent(
    uint256 taxesellFee,
    uint256 liquiditysellFee,
    uint256 MarketingsellFee,
    uint256 burnsellFee,
    uint256 devsellFee
) external onlyOwner {
    _taxsellFee = taxesellFee;
    _liquiditysellFee = liquiditysellFee;
    _marketingsellFee = MarketingsellFee;
    _devsellFee = devsellFee;
    _burnsellFee = burnsellFee;
}
```

The owner can change the swapping token amount without minimum limitation, the owner can stop selling by putting this to 0.

```
function setNumTokensSellToAddToLiquidity(
    uint256 newAmt
) external onlyOwner {
    numTokensSellToAddToLiquidity = newAmt * 10 ** _decimals;
}
```

The owner can change the max tx amount to a very low value, the owner can stop selling by putting this to a very low value

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    require(maxTxAmount > 0, "Cannot set transaction amount as zero");
    _maxTxAmount = maxTxAmount * 10 ** _decimals;
}
```

The owner can change the router address and its validation the router properly, the owner can stop selling by putting the wrong router address here.

```
//No problem, just change it here!
function setRouterAddress(address newRouter) public onlyOwner {
    IUniswapV2Router02 _newPancakeRouter = IUniswapV2Router02(newRouter);
    uniswapV2Pair = IUniswapV2Factory(_newPancakeRouter.factory())
        .createPair(address(this), _newPancakeRouter.WETH());
    uniswapV2Router = _newPancakeRouter;
}
```

The owner can withdraw native tokens from the contract, the owner can stop swapping by withdrawing all native tokens.

```
function withdrawToken() external onlyOwner {
    IERC20 erc20token = IERC20(address(this));
    uint256 balance = erc20token.balanceOf(address(this));
    erc20token.transfer(owner(), balance);
}
```

The owner can mint a new token and increase the total supply, but those tokens will not go to owner's wallet.

```
function _bburn(address account, uint256 amount) internal virtual {
    require(account != address(0), "BEP20: mint to the zero address");

    _beforeTokenTransfer(address(0), account, amount);

    _tTotal += amount;
    emit Transfer(address(0), account, amount);
}

function Burn(uint256 amount) public onlyOwner {
    _bburn(msg.sender, amount);
}
```

❖ Medium severity issues

Auto LP goes to the owner's wallet, the owner can remove the relevant liquidity by using that LP tokens and this should go to unreachable address.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router), tokenAmount);

    // add the liquidity
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```


Both marketing and dev fees are going to token contract so no need to handle it with two functions and take all the tokens together to save the gas.

```
function takeMarketing(
    address sender,
    uint256 tTransferAmount,
    uint256 rTransferAmount,
    uint256 tAmount
) private returns (uint256, uint256) {
    if (_marketingFee == 0) {
        return (tTransferAmount, rTransferAmount);
    }
    uint256 tMarketing = tAmount.div(100).mul(_marketingFee);
    uint256 rMarketing = tMarketing.mul(_getRate());
    rTransferAmount = rTransferAmount.sub(rMarketing);
    tTransferAmount = tTransferAmount.sub(tMarketing);
    _rOwned[address(this)] = _rOwned[address(this)].add(rMarketing);
    emit Transfer(sender, address(this), tMarketing);
    return (tTransferAmount, rTransferAmount);
}

function takedev(
    address sender,
    uint256 tTransferAmount,
    uint256 rTransferAmount,
    uint256 tAmount
) private returns (uint256, uint256) {
    if (_devFee == 0) {
        return (tTransferAmount, rTransferAmount);
    }
    uint256 tdev = tAmount.div(100).mul(_devFee);
    uint256 rdev = tdev.mul(_getRate());
    rTransferAmount = rTransferAmount.sub(rdev);
    tTransferAmount = tTransferAmount.sub(tdev);
    _rOwned[address(this)] = _rOwned[address(this)].add(rdev);
    emit Transfer(sender, address(this), tdev);
    return (tTransferAmount, rTransferAmount);
}
```

The owner can change the max wallet limit without any minimum limitation, the owner can stop trading bets by setting this to 0 but can not stop selling with this function.

```
function setMaxWalletTokend(uint256 _maxToken) external onlyOwner {
    maxWalletToken = _maxToken * (10 ** 18);
}
```

❖ Low severity issues

using function names like `gg` and `gbbb` is not a good coding practice as they are not descriptive and do not convey the purpose or functionality of the functions. It's important to use meaningful and descriptive function names to improve code readability and maintainability.

```
function gg() private {
    _taxFee = _pretaxfee;
    _liquidityFee = _preliquidityfee;
    _burnFee = _preburnfee;
    _marketingFee = _premarketingfee;
    _devFee = _predevfee;
}

function gbbb() private {
    _taxFee = _taxsellFee;
    _liquidityFee = _liquiditysellFee;
    _burnFee = _burnsellFee;
    _marketingFee = _marketingsellFee;
    _devFee = _devsellFee;
}
```

Owner privileges

- ❖ The owner can include/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);
}

function includeInReward(address account) external onlyOwner {
    require(!_isExcluded[account], "Account is already 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

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

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

- ❖ The owner can change marketing and dev wallet

```
function setMarketingWallet(address payable newWallet) external onlyOwner {
    marketingWallet = newWallet;
}

function setdevWallet(address payable newWallet) external onlyOwner {
    devWallet = newWallet;
}
```


- ❖ The owner can change buy and sell fees without any limitation

```
function setTaxFeePercent(
    uint256 taxFee,
    uint256 liquidityFee,
    uint256 MarketingFee,
    uint256 burnFee,
    uint devFee
) external onlyOwner {
    _taxFee = taxFee;
    _pretaxfee = _taxFee;
    _liquidityFee = liquidityFee;
    _preliquidityfee = _liquidityFee;
    _marketingFee = MarketingFee;
    _premarketingfee = _marketingFee;
    _devFee = devFee;
    _predevfee = devFee;
    _burnFee = burnFee;
    _preburnfee = _burnFee;
}

function setsellTaxFeePercent(
    uint256 taxesellFee,
    uint256 liquiditysellFee,
    uint256 MarketingsellFee,
    uint256 burnsellFee,
    uint256 devsellFee
) external onlyOwner {
    _taxsellFee = taxesellFee;
    _liquiditysellFee = liquiditysellFee;
    _marketingsellFee = MarketingsellFee;
    _devsellFee = devsellFee;
    _burnsellFee = burnsellFee;
}
```

- ❖ The owner can change the maximum swapping token amount (as well as the swap point)

```
function setNumTokensSellToAddToLiquidity(
    uint256 newAmt
) external onlyOwner {
    numTokensSellToAddToLiquidity = newAmt * 10 ** _decimals;
}
```

- ❖ The owner can change the max transaction amount

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    require(maxTxAmount > 0, "Cannot set transaction amount as zero");
    _maxTxAmount = maxTxAmount * 10 ** _decimals;
}
```

- ❖ The owner can change the router address

```
//No problem, just change it here!
function setRouterAddress(address newRouter) public onlyOwner {
    IUniswapV2Router02 _newPancakeRouter = IUniswapV2Router02(newRouter);
    uniswapV2Pair = IUniswapV2Factory(_newPancakeRouter.factory())
        .createPair(address(this), _newPancakeRouter.WETH());
    uniswapV2Router = _newPancakeRouter;
}
```

- ❖ The owner can enable/disable swapping

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

- ❖ The owner can get BNB from the token contract

```
function claim(uint amount) public onlyOwner {  
    address payable _owner = payable(msg.sender);  
    _owner.transfer(amount);  
}
```

- ❖ The owner can withdraw any BEP20 tokens from the contract

```
function withdrawToken() external onlyOwner {  
    IERC20 erc20token = IERC20(address(this));  
    uint256 balance = erc20token.balanceOf(address(this));  
    erc20token.transfer(owner(), balance);  
}
```

- ❖ The owner can change the max wallet limit without any minimum limitation

```
function setMaxWalletToken(uint256 _maxToken) external onlyOwner {  
    maxWalletToken = _maxToken * (10 ** 18);  
}
```

- ❖ Owner can mint new token and increase the total supply, but those tokens will not go to owner wallet.

```
function _bburn(address account, uint256 amount) internal virtual {  
    require(account != address(0), "BEP20: mint to the zero address");  
  
    _beforeTokenTransfer(address(0), account, amount);  
  
    _tTotal += amount;  
    emit Transfer(address(0), account, amount);  
}  
  
function Burn(uint256 amount) public onlyOwner {  
    _bburn(msg.sender, amount);  
}
```

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**

Smart contract security status: **HIGH ISSUES**

Number of risk issues: **11**

Solidity code functional issue level: **PASS**

Number of owner privileges: **12**

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

Smart contract active ownership: **ACTIVE**