



RugFreeCoins Audit



EFT Fan Token

Smart Contract Security Audit

March 18, 2022

Contents

Audit details	1
Disclaimer	2
Background	3
About the project	4
Target market and the concept	8
Potential to grow with score points	11
Total Points	11
Contract details	12
Contract code function details	14
Contract description table	16
Security issue checking status	30
Owner privileges	31
Audit conclusion	39

Audit details



Audited project
EFT Fan Token



Contract Address
0x2864c443A1794a70e8ec37aa2C3dD084aB1Eb9eA



Client contact
EFT Fan Team



Blockchain
Binance smart chain



Project website
<https://eftfan.club/>

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 Team to perform an audit of the smart contract.

<https://bscscan.com/token/0x2864c443A1794a70e8ec37aa2C3dD084aB1Eb9eA>

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 improving the security posture of the smart contract by remediating the issues that were identified.

.

About the project

EFT FAN TOKEN (EFF) is a group of investors inspired by the ETH Fan Token Ecosystem (EFT). It's a Token for all those who love the notion of earning Ethereum (ETH) on the Binance Smart Chain (BSC) BEP-20. Binance-Pegged Ethereum (B-ETH) has all the price action and price impacts of ERC-20 ETH but can be transacted with a negligible amount of gas fees on the BSC. This is the epitome of intellectual utilization of blockchain successfully deployed by ETH Fan Token Ecosystem (EFT).

As a fan token, EFF will be supporting EFT with tokenomics that will buyback-reward EFT and buyback-burn EFT. In addition, there will be a weekly lottery pool with B-ETH as the jackpot for lucky long-term holders.

EFF is a crypto product born to serve its purpose as a Fan Token of EFT. It is solely formed to help promote EFT. There will be no doubt that both EFT and non-EFT holders who invest with EFF will enjoy the privileges of EFT's Infinite-Compounding-Loop token mechanism precisely because every holder of EFF will be rewarded with EFT.

Features

- The **EFT Fan rewards** will be distributed in ETF tokens among every holder proportional to how many tokens each individual holds in values of **4% when buying and selling**.
- The **sustainability fee of 3% when buying and selling for marketing** is what allows EFT Fan Token to hold the aforementioned promise. Tokens will be swapped into ETH and will be sent to a marketing wallet. This way, EFT Fan Token will have enough funds to promote the coin and spend for future development and marketing without selling tokens as the traditional way.
- **The fee of 1% when buying and selling** is used for buyback ETF tokens and burn to sustain the ETH Fan Token Ecosystem.
- The additional component included under the sustainability section is a **liquidity fee of 2% when buying and selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

- **The lottery fee is 1% when buying and selling** is what allows EFT Fan Token to become the most commonly known and recognized lottery token in the crypto sphere. In order for a cryptocurrency to grow and gain traction, especially in the Altcoin market, it must have a 'use-case', which only usually comes with the promise of a better future. The EFT Fan Token team is motivated by the idea that the coin will have a use-case from day one! The gambling industry has been around for centuries, and there will be an ever-growing crowd of 'gamblers and players in the crypto sphere, as cryptocurrencies slowly become the staple in terms of money transactions around the world. In order to support this transition, EFT Fan Token wishes to establish itself as the most competitive and well-known lottery token in the industry, all the while progressively growing a following.

With EFT Fan Token, the chances of winning are relative to how many tokens you hold, which means that all holders are incentivized to buy more tokens in the long term if they wish to increase their chances of winning the lottery.

And later the platform will be enhanced to buying a ticket to entry with EFT Fan Token. The EFT Fan Token is sent to the burn wallet and contributes to the daily volume. Users can win by having from 2 Numbers to 6 Numbers correct in sequence. All funds not distributed compounds to the next week's Lotto jackpot growing the Jackpot to a Life-Changing Pot in ETH

- EFT Fan Token has the burn strategy that a **1% fee in each transaction when buying and selling** is getting charged that benefits and rewards those who invest long-term. This feature slowly reduces supply making each EFT Fan Token more and more valuable.

ROADMAP

Q1 2022

- Smart Contract Development and Deployment on BEP-20
- Smart Contract Audit Completion
- Website and Social Media Unveiling
- Telegram Community Building
- Private and Pre-sale
- Pancake Swap Launch
- Coin Market Cap and Coin Gecko Listing
- Lottery D-app
- Reward Dashboard and Calculator

Q2 2022

- Growing Telegram Community
- Influencers Marketing Campaigns
- Partnership with other Projects
- Staking pool
- Start P2E development
- CEX Listing

Q3 2022

- P2E Launch
- Additional CEX
- NFT Marketplace
- NFT Staking

Tokenomics

12% fee when selling

- 4% of trade goes to holders pockets in EFT token rewards.
- 3% of trade goes to the marketing wallet in ETH
- 2% of trade goes to the liquidity pool
- 1% of trade goes to the buyback and burn of EFT.
- 1% of trade goes to the burn wallet for EFF burn.
- 1% of trade goes to the lottery pool in ETH.

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 is ready to hold and be eligible to win in the daily lottery
- Anyone who is ready to hold a large portion of tokens and be eligible to get a high chance of winning in the weekly lottery.
- Anyone who's interested in taking part with staking.
- Anyone who's interested in taking part in decision-making.
- Anyone who's interested in taking part with the future plans of the EFT Fan token.
- Anyone who's interested in making financial transactions with any other party using ETH, ETH Fan token and EFT Fan token as the currency.

Core concept

The EFT Fan reward system

4% of each transaction when buying and selling gets converted to EFT tokens and is split amongst all holders. Holders will be eligible to receive tokens in each transaction and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The **sustainability fee of 3% when buying and selling for dev and marketing** is what allows EFT Fan token to promote the token and use funds to further the development of the platform. Tokens will be swapped into ETH and will be sent to a marketing wallet. This way, EFT Fan will have access to the funds without selling tokens as the traditional way, which will enable them to consume funds without hurting the project.

The liquidity fee of 2% when buying and selling, is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

The fee of 1% when buying and selling is used for buyback ETF tokens and burn to sustain the ETH Fan Token Ecosystem.

Lottery pool & lottery platform

The concept encourages,

- Investors, to hold the token for a long time, which makes them believe in the project and keep the hopes high of expecting to win a huge prize at once.
- Investors buy more and more since the chance of winning is higher.
- The concept is revolutionary and certainly can get the attraction of new investors as the project progresses along.
- Project market price and market cap can keep stable if everything goes according to the plan since keeping tokens will seem more profitable than selling.

Weekly Lottery drawing

EFT Fan lottery is with a strong use case specifically targeting the gambling industry aiming for any long-term believers and holders to give a chance to be eligible for the weekly lottery and win. The most unique core part of the EFT Fan is that the chances of winning are relative to how many tokens investors hold, which means that all holders are incentivized to buy more tokens in the long term if they wish to increase their chances of winning the lottery.

How Chances of Winning are Calculated

Chances of winning will be calculated in indirect proportion to how many tokens each holder has. This means that having more tokens does increase your chances of winning, but not in a linear fashion. Instead, a logarithmic function will be used to convert the proportion of holdings that each investor has, and calculate their chances of winning accordingly. This will lower the discrepancy in the probability of winning between a whale and a small investor while keeping our largest investors at an advantage.

No. of tokens	% chance of winning	Log transformation	% of winning (log transformation)
15	0.50	1.17609	0.36784
07	0.23	0.84510	0.26432
05	0.17	0.69897	0.21861
03	0.10	0.477120	0.14923
Total		3.19728	1

Lottery Drawing Dapp

The lottery platform will be visible in an interface, where all contract holders are visible with their wallet IDs and the number of tokens they hold. Holders can connect wallets and check the probability of winning against the rest of the holders.

Winners will be chosen on a random draw, live on video chat. The winners will be populated on the web with wallet IDs and the amounts they won.

Potential to grow with score points

1.	Project efficiency	10/10
2.	Project uniqueness	10/10
3	Information quality	10/10
4	Service quality	10/10
5	System quality	10/10
6	Impact on the community	10/10
7	Impact on the business	10/10
8	Preparing for the future	9/10
Total Points		9.875/10

Contract details

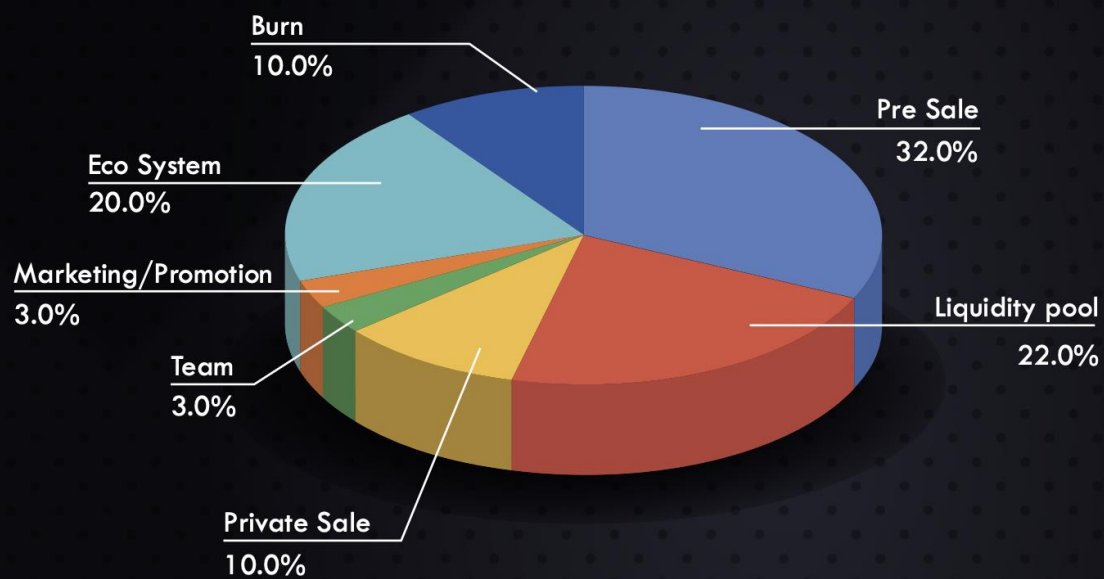
Token contract details for 18th March 2022

Contract name	EFT Fan Token
Contract address	0x2864c443A1794a70e8ec37aa2C3dD084aB1Eb9eA
Token supply	100,000,000,000
Token ticker	\$EFF
Decimals	9
Token holders	3
Transaction count	5
Dividend tracker	0x9af7a61ceb9d5bf1d4ba7cb9aa8c611c75367b6c
Lottery wallet	0xeffe48e632c21482c462e92ad3625f0a099e3c22
Marketing wallet	0xabd50d22a9665e7ada7e7799d40e126c164f5c15
Game wallet	0x920ad869b16b8a569b8980179f2e7850f447b56a
Contract deployer address	0x2550bcA4990892B694f4A9B5432110F0af433CCb
Contract's current owner address	0x1a24a36895fc908b08ca2df200e2d52ee1f04646

Tokens are distributed as follows:

EFT Fan Token Earn is deployed on the Binance Smart Chain (BSC) with an initial supply of 100,000,000,000 (100 Billion) tokens.

in %








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	Access control of owners	High
		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














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
IUniswapV2Factory	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 !
IUniswapV2Router01	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	swapExactTokensForETHSupporting FeeOnTransferTokens	External !		NO !
SafeMath	Library			
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 		
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 !
IDividendDistributor	Interface			
L	setDistributionCriteria	External !		NO !
L	setShare	External !		NO !
L	deposit	External !		NO !
L	process	External !		NO !
L	purge	External !		NO !
DividendDistributor	Implementation	IDividendDistributor		
L		Public !		NO !
L		External !		NO !
L	setDistributionCriteria	External !		onlyToken
L	purge	External !		onlyToken
L	setShare	External !		onlyToken
L	deposit	External !		onlyToken
L	process	External !		onlyToken
L	shouldDistribute	Internal 		
L	distributeDividend	Internal 		









































L	claimDividend	External !		NO !
L	getUnpaidEarnings	Public !		NO !
L	getHolderDetails	Public !		NO !
L	getCumulativeDividends	Internal 		
L	getLastProcessedIndex	External !		NO !
L	getNumberOfTokenHolders	External !		NO !
L	getShareHoldersList	External !		NO !
L	totalDistributedRewards	External !		NO !
L	addShareholder	Internal 		
L	removeShareholder	Internal 		
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	_transferOwnership	Internal 		






















EFF	Implementation	IERC20, Ownable		
L		Public !		NO !
L		External !		NO !
L	totalSupply	External !		NO !
L	name	Public !		NO !
L	symbol	Public !		NO !
L	decimals	Public !		NO !
L	balanceOf	Public !		NO !
L	getHolderDetails	Public !		NO !
L	getLastProcessedIndex	Public !		NO !
L	getNumberOfTokenHolders	Public !		NO !
L	totalDistributedRewards	Public !		NO !
L	allowance	External !		NO !
L	approve	Public !		NO !
L	_approve	Internal 		
L	approveMax	External !		NO !
L	transfer	External !		NO !
L	transferFrom	External !		NO !
L	_transferFrom	Internal 		
L	_basicTransfer	Internal 		
L	shouldTakeFee	Internal 		





















L	takeFee	Internal 		
L	shouldSwapBack	Internal 		
L	clearStuckBalance	External 		onlyOwner
L	getBep20Tokens	External 		onlyOwner
L	updateBuyFees	Public 		onlyOwner
L	updateSellFees	Public 		onlyOwner
L	updateSwapPercentages	Public 		onlyOwner
L	enableTrading	Public 		onlyOwner
L	whitelistPreSale	Public 		onlyOwner
L	___claimRewards	Public 		NO 
L	claimProcess	Public 		NO 
L	blackListWallets	Public 		onlyOwner
L	isBlacklisted	Public 		NO 
L	isRewardExclude	Public 		NO 
L	isFeeExclude	Public 		NO 
L	isMaxWalletExclude	Public 		NO 
L	isMaxTxExcluded	Public 		NO 
L	setIsMaxTxExempt	External 		onlyOwner
L	setMaxTxAmount	External 		onlyOwner
L	isExemptTimeLock	Public 		NO 
L	changeSellCoolDownTime	Public 		onlyOwner

















L	enableSellCollDown	Public !		onlyOwner
L	exemptTimeLock	Public !		onlyOwner
L	swapBackInBnb	Internal 		swapping
L	swapAndLiquify	Private 		
L	swapTokensForEth	Private 		
L	swapTokensForTokens	Private 		
L	addLiquidity	Private 		
L	setIsDividendExempt	External !		onlyOwner
L	setIsFeeExempt	External !		onlyOwner
L	setIsMaxWalletExempt	External !		onlyOwner
L	addAuthorizedWallets	External !		onlyOwner
L	setMarketingWallet	External !		onlyOwner
L	setLotteryWallet	External !		onlyOwner
L	setGamePoolAddress	External !		onlyOwner
L	setStakePoolAddress	External !		onlyOwner
L	changeBuyBackAndBurnToken	External !		onlyOwner
L	changeLotteryAndMarketingToken	External !		onlyOwner
L	setMaxWalletToken	External !		onlyOwner
L	changeSellFeeMultiplier	External !		onlyOwner
L	setSwapBackSettings	External !		onlyOwner
L	setDistributionCriteria	External !		onlyOwner

L	setDistributorSettings	External !		onlyOwner
L	purgeBeforeSwitch	Public !		onlyOwner
L	includeMeinRewards	Public !		NO !
L	switchToken	Public !		onlyOwner
IDividendDistributor	Interface			
L	setDistributionCriteria	External !		NO !
L	setShare	External !		NO !
L	deposit	External !		NO !
L	process	External !		NO !
L	purge	External !		NO !
DividendDistributor	Implementation	IDividendDistributor		
L		Public !		NO !
L		External !		NO !
L	setDistributionCriteria	External !		onlyToken
L	purge	External !		onlyToken
L	setShare	External !		onlyToken
L	deposit	External !		onlyToken
L	process	External !		onlyToken
L	shouldDistribute	Internal 		



L	distributeDividend	Internal 		
L	claimDividend	External 		NO 
L	getUnpaidEarnings	Public 		NO 
L	getHolderDetails	Public 		NO 
L	getCumulativeDividends	Internal 		
L	getLastProcessedIndex	External 		NO 
L	getNumberOfTokenHolders	External 		NO 
L	getShareholdersList	External 		NO 
L	totalDistributedRewards	External 		NO 
L	addShareholder	Internal 		
L	removeShareholder	Internal 		
EFF	Implementation	IERC20, Ownable		
L		Public 		NO 
L		External 		NO 
L	totalSupply	External 		NO 
L	name	Public 		NO 
L	symbol	Public 		NO 
L	decimals	Public 		NO 
L	balanceOf	Public 		NO 
L	getHolderDetails	Public 		NO 

L	getLastProcessedIndex	Public !		NO !
L	getNumberOfTokenHolders	Public !		NO !
L	totalDistributedRewards	Public !		NO !
L	allowance	External !		NO !
L	approve	Public !		NO !
L	_approve	Internal 		
L	approveMax	External !		NO !
L	transfer	External !		NO !
L	transferFrom	External !		NO !
L	_transferFrom	Internal 		
L	_basicTransfer	Internal 		
L	shouldTakeFee	Internal 		
L	takeFee	Internal 		
L	shouldSwapBack	Internal 		
L	clearStuckBalance	External !		onlyOwner
L	getBep20Tokens	External !		onlyOwner
L	updateBuyFees	Public !		onlyOwner
L	updateSellFees	Public !		onlyOwner
L	updateSwapPercentages	Public !		onlyOwner
L	enableTrading	Public !		onlyOwner
L	whitelistPreSale	Public !		onlyOwner

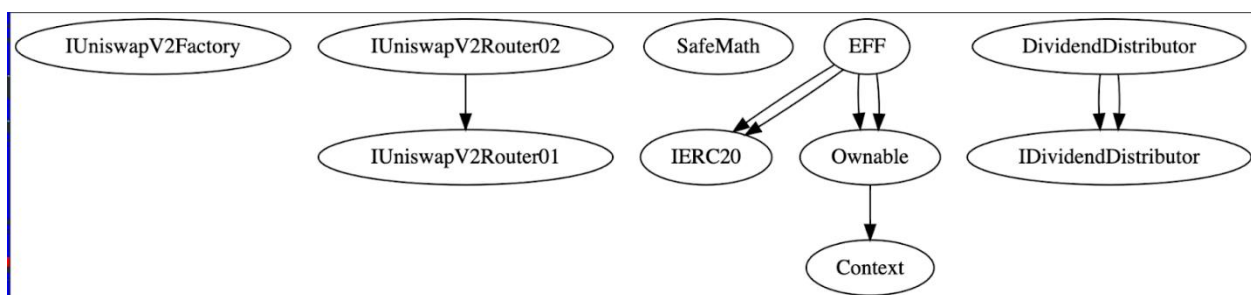
L	__claimRewards	Public !		NO!
L	claimProcess	Public !		NO!
L	blackListWallets	Public !		onlyOwner
L	isBlacklisted	Public !		NO!
L	isRewardExclude	Public !		NO!
L	isFeeExclude	Public !		NO!
L	isMaxWalletExclude	Public !		NO!
L	isMaxTxExcluded	Public !		NO!
L	setIsMaxTxExempt	External !		onlyOwner
L	setMaxTxAmount	External !		onlyOwner
L	isExemptTimeLock	Public !		NO!
L	changeSellCoolDownTime	Public !		onlyOwner
L	enableSellCollDown	Public !		onlyOwner
L	exemptTimeLock	Public !		onlyOwner
L	swapBackInBnb	Internal 		swapping
L	swapAndLiquify	Private 		
L	swapTokensForEth	Private 		
L	swapTokensForTokens	Private 		
L	addLiquidity	Private 		
L	setIsDividendExempt	External !		onlyOwner
L	setIsFeeExempt	External !		onlyOwner

L	setIsMaxWalletExempt	External !		onlyOwner
L	addAuthorizedWallets	External !		onlyOwner
L	setMarketingWallet	External !		onlyOwner
L	setLotteryWallet	External !		onlyOwner
L	setGamePoolAddress	External !		onlyOwner
L	setStakePoolAddress	External !		onlyOwner
L	changeBuyBackAndBurnToken	External !		onlyOwner
L	changeLotteryAndMarketingToken	External !		onlyOwner
L	setMaxWalletToken	External !		onlyOwner
L	changeSellFeeMultiplier	External !		onlyOwner
L	setSwapBackSettings	External !		onlyOwner
L	setDistributionCriteria	External !		onlyOwner
L	setDistributorSettings	External !		onlyOwner
L	purgeBeforeSwitch	Public !		onlyOwner
L	includeMeinRewards	Public !		NO!
L	switchToken	Public !		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

No low severity issues found

❖ Centralization risk

High severity issues

- ❖ The owner can change sell fee multiplier without any maximum limitation, the owner can set 100% or more sell tax using this function. (Only if selling tokens within sell cool downtime after buying)

```
ftrace | funcSig
function changeSellFeeMultiplier(uint256 amount↑) external onlyOwner {
    sellTaxMultiplier = amount↑;
}
```

- ❖ The owner can change the max transaction amount without any minimum limit

```
ftrace | funcSig
function setMaxTxAmount(uint256 amount↑) external onlyOwner {
    maxTxAmount = amount↑ * (10**9);
}
```

- ❖ The owner can enable/disable trading

```
// switch Trading
ftrace | funcSig
function enableTrading(bool _status↑) public onlyOwner {
    tradingOpen = _status↑;
}
```


Owner privileges

- ❖ The owner can get BNB in contract to owner wallet

```
ftrace | funcSig
function clearStuckBalance(uint256 amountPercentage↑) external onlyOwner {
    uint256 amountBNB = address(this).balance;
    payable(msg.sender).transfer((amountBNB * amountPercentage↑) / 100);
}
```

- ❖ The owner can get any BEP20 tokens in the contract to the owner wallet

```
ftrace | funcSig
function getBep20Tokens(address _tokenAddress↑, uint256 amount↑)
    external
    onlyOwner
{
    require(
        IERC20(_tokenAddress↑).balanceOf(address(this)) >= amount↑,
        "No Enough Tokens"
    );
    IERC20(_tokenAddress↑).transfer(msg.sender, amount↑);
}
```

- ❖ The owner can change buy fees maximum up to 25%

```
ftrace | funcSig
function updateBuyFees(
    uint256 reward↑,
    uint256 marketing↑,
    uint256 liquidity↑,
    uint256 burn↑,
    uint256 staking↑,
    uint256 buyBack↑,
    uint256 gamePool↑,
    uint256 lottery↑
) public onlyOwner {
    buyRewardFee = reward↑;
    buyMarketingFee = marketing↑;
    buyLiquidityFee = liquidity↑;
    buyBurnFee = burn↑;
    buyStakePoolFee = staking↑;
    buyBuyBackFee = buyBack↑;
    buyGameFee = gamePool↑;
    buyLotteryFee = lottery↑;
    buyTotalFees = reward↑.add(marketing↑).add(liquidity↑).add(burn↑).add(
        staking↑
    );
    buyTotalFees = buyTotalFees.add(buyBack↑).add(gamePool↑).add(lottery↑);

    require(buyTotalFees <= 25, "Fees can not be greater than 25%");
}
```

- ❖ The owner can change sell fees maximum up to 25%

```
ftrace | funcSig
function updateSellFees(
    uint256 reward↑,
    uint256 marketing↑,
    uint256 liquidity↑,
    uint256 burn↑,
    uint256 staking↑,
    uint256 buyBack↑,
    uint256 gamePool↑,
    uint256 lottery↑
) public onlyOwner {
    sellRewardFee = reward↑;
    sellMarketingFee = marketing↑;
    sellLiquidityFee = liquidity↑;
    sellBurnFee = burn↑;
    sellStakePoolFee = staking↑;
    sellBuyBackFee = buyBack↑;
    sellGameFee = gamePool↑;
    sellLotteryFee = lottery↑;
    sellTotalFees = reward↑.add(marketing↑).add(liquidity↑).add(burn↑).add(
        staking↑
    );
    sellTotalFees = buyTotalFees.add(buyBack↑).add(gamePool↑).add(lottery↑);

    require(sellTotalFees <= 25, "Fees can not be greater than 25%");
}
```

- ❖ The owner can change token swap percentages

```
// update swap percentages
ftrace | funcSig
function updateSwapPercentages(
    uint256 reward↑,
    uint256 marketing↑,
    uint256 liquidity↑,
    uint256 burn↑,
    uint256 lottery↑
) public onlyOwner {
    rewardSwap = reward↑;
    marketingSwap = marketing↑;
    liquiditySwap = liquidity↑;
    burnSwap = burn↑;
    lotterySwap = lottery↑;
    totalSwap = reward↑.add(marketing↑).add(liquidity↑).add(burn↑).add(lottery↑);
}

// switch Trading
```

- ❖ The owner can enable/disable trading

```
// switch Trading
ftrace | funcSig
function enableTrading(bool _status↑) public onlyOwner {
    tradingOpen = _status↑;
}
```

- ❖ The owner can whitelist presale address

```
ftrace | funcSig
function whitelistPreSale(address _preSale↑) public onlyOwner {
    isFeeExempt[_preSale↑] = true;
    isDividendExempt[_preSale↑] = true;
    isAuthorized[_preSale↑] = true;
    isMaxWalletExempt[_preSale↑] = true;
}
```

- ❖ The owner can add/remove wallets from blacklist

```
ftrace | funcSig
function blacklistWallets(address wallet↑, bool _status↑) public onlyOwner {
    isBlacklist[wallet↑] = _status↑;
}
```

- ❖ The owner can include/exclude wallets from max transaction

```
ftrace | funcSig
function setIsMaxTxExempt(address holder↑, bool exempt↑) external onlyOwner {
    isMaxTxExempt[holder↑] = exempt↑;
}
```

- ❖ The owner can change max transaction amount

```
ftrace | funcSig
function setMaxTxAmount(uint256 amount↑) external onlyOwner {
    maxTxAmount = amount↑ * (10**9);
}
```

- ❖ The owner can change sell cool downtime and enable disable sell cooldown

```
ftrace | funcSig
function changeSellCoolDownTime(uint256 _time↑) public onlyOwner {
    cooldownTimerInterval = _time↑;
}

ftrace | funcSig
function enableSellCollDown(bool _status↑) public onlyOwner {
    coolDownEnabled = _status↑;
}
```

- ❖ The owner can include/exclude wallets from selling cool down

```
ftrace | funcSig
function exemptTimeLock(address wallet↑, bool _status↑) public onlyOwner {
    isTimelockExempt[wallet↑] = _status↑;
}
```

- ❖ The owner can include/exclude wallets from rewards

```
ftrace | funcSig
function setIsDividendExempt(address holder↑, bool exempt↑)
  external
  onlyOwner
{
  require(holder↑ != address(this) && holder↑ != pair);
  isDividendExempt[holder↑] = exempt↑;
  if (exempt↑) {
    dividendTracker.setShare(holder↑, 0);
  } else {
    dividendTracker.setShare(holder↑, _balances[holder↑]);
  }
}
```

- ❖ The owner can include/exclude wallets from fee

```
ftrace | funcSig
function setIsFeeExempt(address holder↑, bool exempt↑) external onlyOwner {
  isFeeExempt[holder↑] = exempt↑;
}
```

- ❖ The owner can include/exclude wallets from max wallet

```
ftrace | funcSig
function setIsMaxWalletExempt(address holder↑, bool exempt↑)
  external
  onlyOwner
{
  isMaxWalletExempt[holder↑] = exempt↑;
}
```

- ❖ The owner can add/remove authorized wallets

```
ftrace | funcSig
function addAuthorizedWallets(address holder↑, bool exempt↑)
  external
  onlyOwner
{
  isAuthorized[holder↑] = exempt↑;
}
```

- ❖ The owner can change all fee receiver address

```
ftrace | funcSig
function setMarketingWallet(address _marketingFeeReceiver↑)
  external
  onlyOwner
{
  marketingFeeReceiver = _marketingFeeReceiver↑;
}

ftrace | funcSig
function setLotteryWallet(address _lotteryFeeReceiver↑) external onlyOwner {
  lotteryFeeReceiver = _lotteryFeeReceiver↑;
}

ftrace | funcSig
function setGamePoolAddress(address _gameWallet↑) external onlyOwner {
  gameWallet = _gameWallet↑;
}

ftrace | funcSig
function setStakePoolAddress(address _stakePool↑) external onlyOwner {
  stakePoolAddress = _stakePool↑;
}
```

- ❖ The owner can change buy back and burn token address and marketing and lottery fee receive token

```
ftrace | funcSig
function changeBuyBackAndBurnToken(address _tokenAddress↑)
  external
  onlyOwner
{
  BURN = _tokenAddress↑;
}

ftrace | funcSig
function changeLotteryAndMarketingToken(address _tokenAddress↑)
  external
  onlyOwner
{
  LOTTERY = _tokenAddress↑;
}
```


- ❖ The owner can change max wallet token

```
ftrace | funcSig
function setMaxWalletToken(uint256 amount↑) external onlyOwner {
    maxWalletTokens = amount↑ * (10**9);
}
```

- ❖ The owner can change sell fee multiplier

```
ftrace | funcSig
function changeSellFeeMultiplier(uint256 amount↑) external onlyOwner {
    sellTaxMultiplier = amount↑;
}
```

- ❖ The owner can enable/disable swap back and can change swap point

```
ftrace | funcSig
function setSwapBackSettings(bool _enabled↑, uint256 _amount↑)
    external
    onlyOwner
{
    swapEnabled = _enabled↑;
    swapThreshold = _amount↑;
}
```

- ❖ The owner can change the minimum reward period and can change the minimum reward distribute amount

```
ftrace | funcSig
function setDistributionCriteria(
    uint256 _minPeriod↑,
    uint256 _minDistribution↑
) external onlyOwner {
    dividendTracker.setDistributionCriteria(_minPeriod↑, _minDistribution↑);
}
```

- ❖ The owner can get tokens in the reward tracker before changing the reward address

```
ftrace | funcSig
function purgeBeforeSwitch() public onlyOwner {
    dividendTracker.purge(msg.sender);
}
```

- ❖ The owner can change the reward token address

```
ftrace | funcSig
function switchToken(address rewardToken↑, bool isIncludeHolders↑)
    public
    onlyOwner
{
    require(rewardToken↑ != WBNB, "Can not reward BNB in this tracker");
    REWARD = rewardToken↑;
    // get current shareholders list
    address[] memory currentHolders = dividendTracker.getShareHoldersList();
    dividendTracker = new DividendDistributor(address(router), rewardToken↑);
    if (isIncludeHolders↑) {
        // add old share holders to new tracker
        for (uint256 i = 0; i < currentHolders.length; i++) {
            try
            {
                dividendTracker.setShare(
                    currentHolders[i],
                    balances[currentHolders[i]]
                )
            }
            catch {}
        }
    }

    emit ChangeRewardTracker(rewardToken↑);
}
```


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: **PASSED**

Number of risk issues: **3**

Solidity code functional issue level: **PASSED**

Number of owner privileges: **24**

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

Smart contract active ownership: **YES**