RugFreeCoins Audit

VERSA

VERSATILE FINANCE

Versa Token

Smart Contract Security Audit

August 24, 2022

Contents

Audit details

Audited project
Versatile Finance Token

Contract Address
0xfD3dBB4709Af9FeEB87EB842Cf6b6b5F37B30fAc

Client contact
Versatile Finance Team

Blockchain
Binance smart chain

Project website
https://versatile.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 the Versa Team to perform an audit of the smart contract.

https://bscscan.com/address/0xfD3dBB4709Af9FeEB87EB842Cf6b6b5F37B30fAc

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

.

About the project

Features

- The **BUSD rewards** will be distributed among every holder proportional to how many tokens each individual holds in values of **3% when buying and selling**.

- The additional component included under the sustainability section is a **liquidity fee of 1% when buying and selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

- The **sustainability fee of 3% for marketing and development and 1% strategic wallet**, is what allows Versa Token to hold the aforementioned promise. Tokens will be swapped into BUSD and will be sent to the marketing wallet and strategic wallet. This way, Versa Token will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

ROADMAP

Crawl

- Contract and Audit
- Business Incubator Platform - Initial Release
- Seed and Pre-sale
- PCS Launch
- Business and Partnership pitch deck
- Core Partnership onboarding

Walk

- Versa Rewards and Autoswap
- Versa Blockchain Fund
- Versa Lottery
- Versa Stake
- Business Incubator Platform - New utilities and Services
- Expand Team and Partnership

Run

- Market Research on DeFi trends
- New utilities ideation and selection
- Business Incubator Platform - New utilities and Services
- Expand Partnership
- Scale Versa Blockchain Fund
- Secret utility planning kick-off

Fly

- Register Versatile Finance (Country TBD)
- Regulatory approval for Versa Blockchain Fund
- Secret utility build and launch
- Continue Building new products and service

Tokenomics

8% fee when buying and selling

- 3% of trade goes in BUSD among all holders
- 3% of trade goes to the marketing & development wallet in BNB.
- 1% of trade goes to the liquidity pool.
- 1% of trade goes to the strategic project tax wallet.

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 future plans of the Versa token.
- Anyone who’s interested in making financial transactions with any other party using Versa Token as the currency.

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	9/10
6	Impact on the community	10/10
7	Impact on the business	10/10
8	Preparing for the future	10/10
9	Smart contract security	10/10
10	Smart contract functionality assessment	10/10
Total Points		**10/10**

Contract details

Token contract details for 24th August 2022

Contract name	Versatile Finance
Contract address	0xfD3dBB4709Af9FeEB87EB842Cf6b6b5F37B30fAc
Token supply	1,000,000,000
Token ticker	$VERSA
Decimals	9
Token holders	3
Transaction count	9
Dev Fee Receiver	0xddb447078428c9bed6287a0cfc80a25120f1724a
Dividend Tracker	0x94f9839cf56565f5dea464cc4a311e0d7205d84d
P2E fee receiver	0x47644e55068c0cd885b31bd7f589e7b1d6a39aeb
Stake Fee Receiver	0x02ce017765da681ba29fbbeecb990515d172e617
Strategic Fee Receiver	0x31e0fadd311c26d1b2ef645533cc03ebd27846fd
Contract deployer address	0xD28EBE5C504cC7416e287881e6991223F353D95f
Contract's current owner address	0x22ca22519a1bed1834933af0ed8aa1f66c4281c2

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
		Self-destruct function security	pass
3	Business security	Access control of owners	pass
		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
VERSA	**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❗

└	_approve	Internal 🔒	🛑	
└	approveMax	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	_transferFrom	Internal 🔒	🛑	
└	_basicTransfer	Internal 🔒	🛑	
└	shouldTakeFee	Internal 🔒		
└	takeFee	Internal 🔒	🛑	
└	shouldSwapBack	Internal 🔒		
└	clearStuckBalance	External ❗	🛑	onlyOwner
└	updateLpAddress	External ❗	🛑	onlyOwner
└	getBep20Tokens	External ❗	🛑	onlyOwner
└	updateBuyFees	Public ❗	🛑	onlyOwner
└	updateSellFees	Public ❗	🛑	onlyOwner
└	updateSwapPercentages	Public ❗	🛑	onlyOwner
└	enableTrading	Public ❗	🛑	onlyOwner
└	whitelistPreSale	Public ❗	🛑	onlyOwner
└	___claimRewards	Public ❗	🛑	NO❗
└	claimProcess	Public ❗	🛑	NO❗
└	isRewardExclude	Public ❗		NO❗
└	isFeeExclude	Public ❗		NO❗

L	isMaxTxExcluded	Public ❗		NO❗
L	isMaxWalletExcluded	Public ❗		NO❗
L	setMaxTxAmount	External ❗	🛑	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	addAuthorizedWallets	External ❗	🛑	onlyOwner
L	setMaxWalletToken	External ❗	🛑	onlyOwner
L	changeFeeWallets	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
L	whitelistCustomToken	Public ❗	🛑	onlyOwner
L	setCustomToken	Public ❗	🛑	NO❗

L	setMaxTxExempt	Public ❗	🛑	onlyOwner
L	setMaxWalletExempt	Public ❗	🛑	onlyOwner
Ownable	**Implementation**	**Context**		
L		Public ❗	🛑	NO❗
L	owner	Public ❗		NO❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	_transferOwnership	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❗
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 🔒		
IUniswapV2 Router02	**Interface**	**IUniswapV2 Router01**		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
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❗
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❗

└	permit	External ❗	🛑	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	mint	External ❗	🛑	NO❗
└	burn	External ❗	🛑	NO❗
└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗
IDividend Distributor	**Interface**			
└	setDistributionCriteria	External ❗	🛑	NO❗
└	setShare	External ❗	🛑	NO❗
└	deposit	External ❗	🛑	NO❗
└	process	External ❗	🛑	NO❗

L	purge	External ❗	🛑	NO❗
Dividend Distributor	**Implementation**	**IDividend Distributor**		
L		Public ❗	🛑	NO❗
L		External ❗	💵	NO❗
L	setDistributionCriteria	External ❗	🛑	onlyToken
L	purge	External ❗	🛑	onlyToken
L	setCustomTokens	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❗

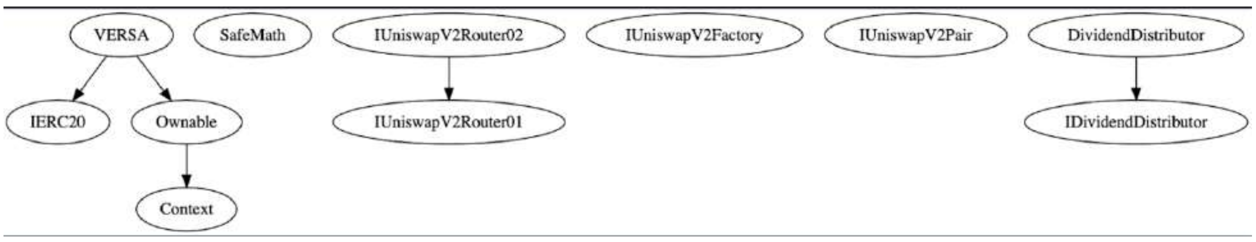
L	addShareholder	Internal 🔒	🛑	
L	removeShareholder	Internal 🔒	🛑	
L	swapTokensForTokens	Private 🔐	🛑	
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
IUniswapV2 Router01	**Interface**			
L	factory	External ❗		NO❗
L	WETH	External ❗		NO❗
L	addLiquidity	External ❗	🛑	NO❗
L	addLiquidityETH	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❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy



Security issue checking status

- **High severity issues**
  No High severity issues found

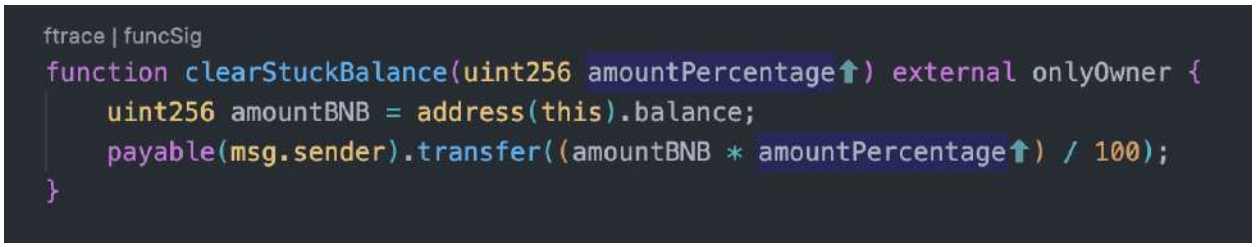
- **Medium severity issues**
  No medium severity issues found

- **Low severity issues**
  No low severity issues found

- **Centralization Risk**
  No centralization issues found

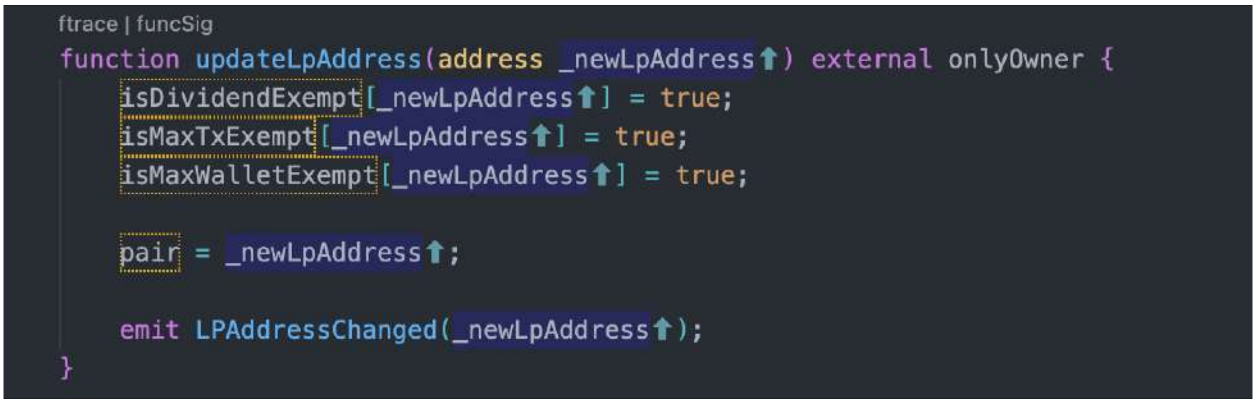
Owner privileges

❖ The owner can get contract BNB balance to owner wallet



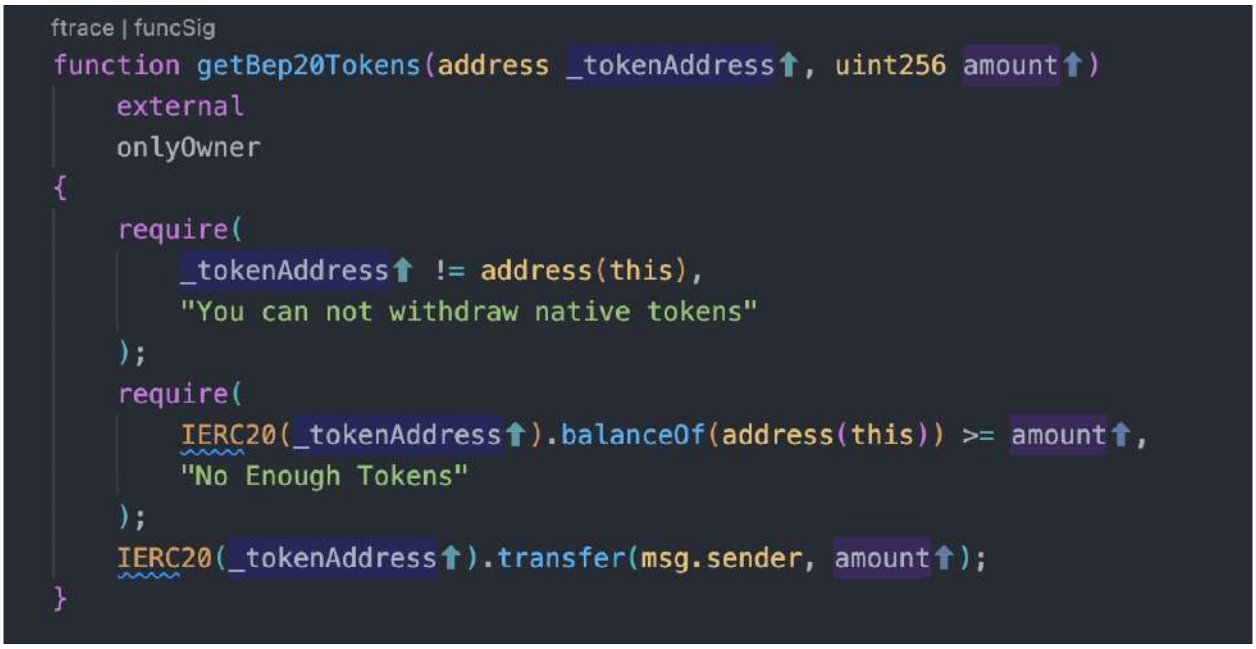
```
function clearStuckBalance(uint256 amountPercentage) external onlyOwner {
    uint256 amountBNB = address(this).balance;
    payable(msg.sender).transfer((amountBNB * amountPercentage) / 100);
}
```

❖ The owner can update the LP address



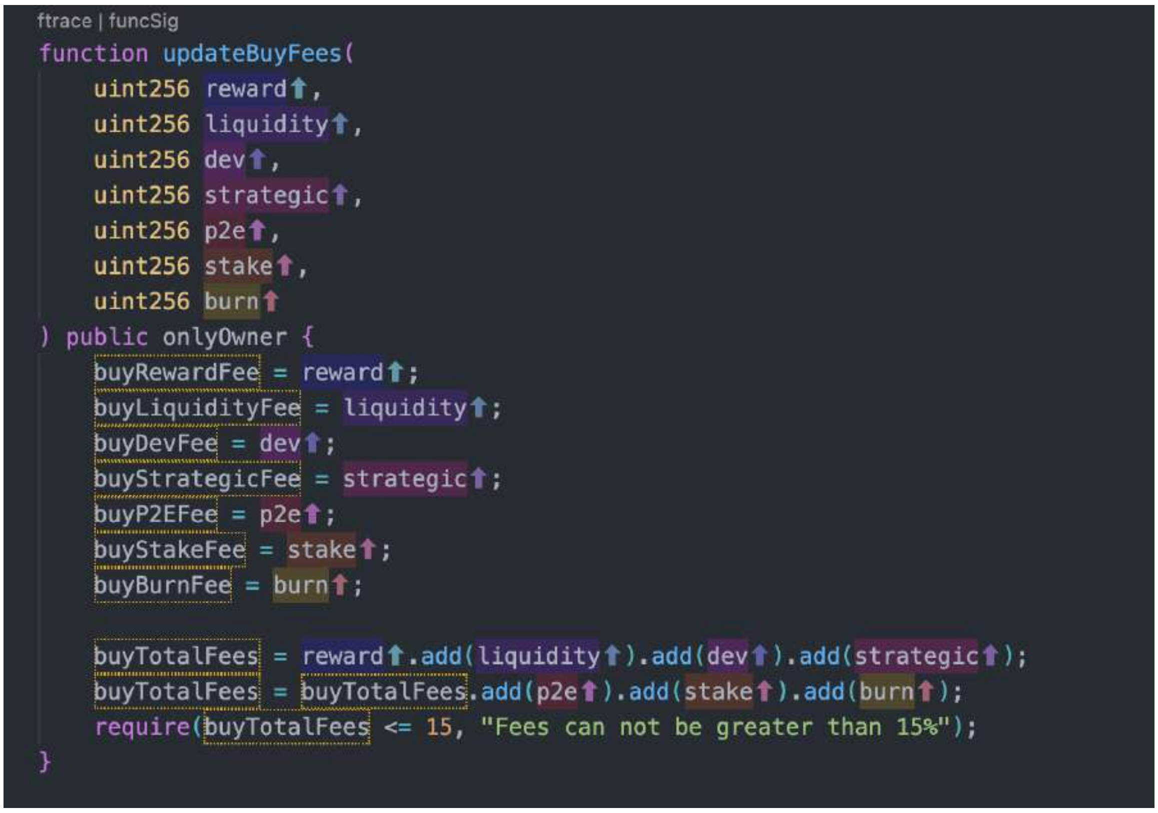
```
function updateLpAddress(address _newLpAddress) external onlyOwner {
    isDividendExempt[_newLpAddress] = true;
    isMaxTxExempt[_newLpAddress] = true;
    isMaxWalletExempt[_newLpAddress] = true;

    pair = _newLpAddress;

    emit LPAddressChanged(_newLpAddress);
}
```

❖ The owner can transfer any BEP20 tokens in contract to any wallet



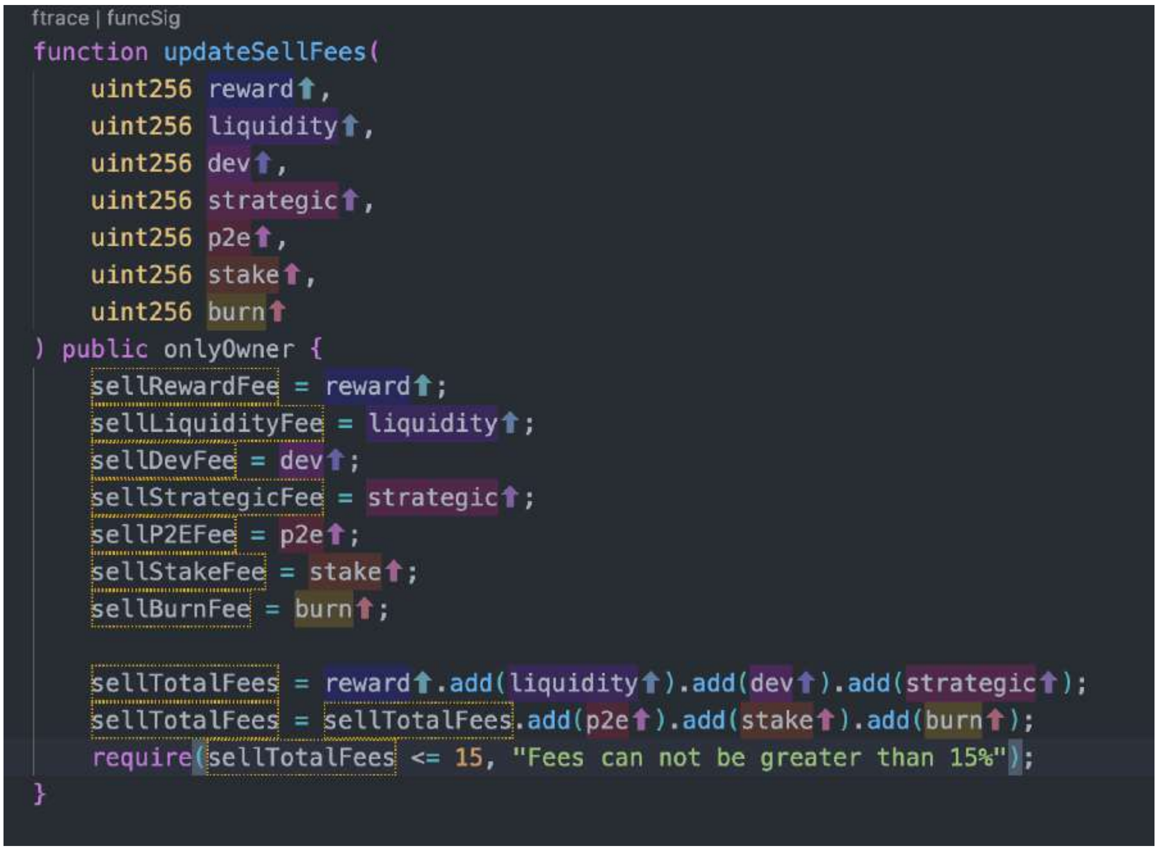
```
function getBep20Tokens(address _tokenAddress, uint256 amount)
    external
    onlyOwner
{
    require(
        _tokenAddress != address(this),
        "You can not withdraw native tokens"
    );
    require(
        IERC20(_tokenAddress).balanceOf(address(this)) >= amount,
        "No Enough Tokens"
    );
    IERC20(_tokenAddress).transfer(msg.sender, amount);
}
```

❖ The owner can update all buy fees maximum up to 15%



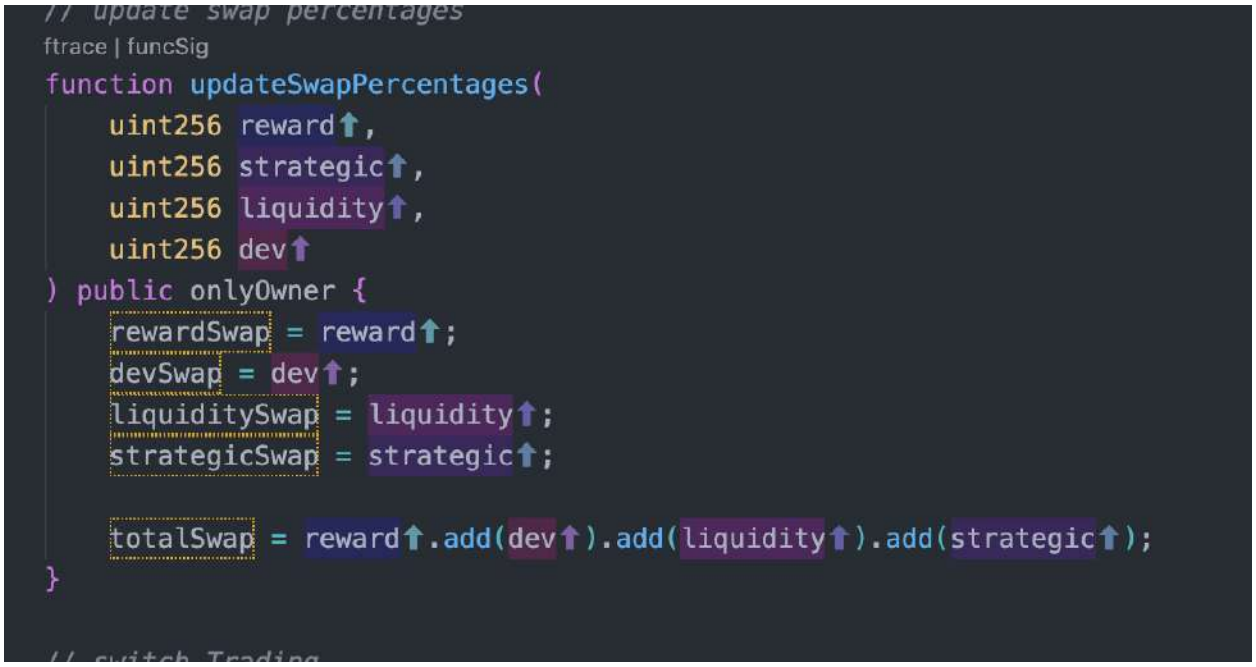
```
ftrace | funcSig
function updateBuyFees(
    uint256 reward,
    uint256 liquidity,
    uint256 dev,
    uint256 strategic,
    uint256 p2e,
    uint256 stake,
    uint256 burn
) public onlyOwner {
    buyRewardFee = reward;
    buyLiquidityFee = liquidity;
    buyDevFee = dev;
    buyStrategicFee = strategic;
    buyP2EFee = p2e;
    buyStakeFee = stake;
    buyBurnFee = burn;

    buyTotalFees = reward.add(liquidity).add(dev).add(strategic);
    buyTotalFees = buyTotalFees.add(p2e).add(stake).add(burn);
    require(buyTotalFees <= 15, "Fees can not be greater than 15%");
}
```

❖ The owner can update sell fees maximum up to 15%



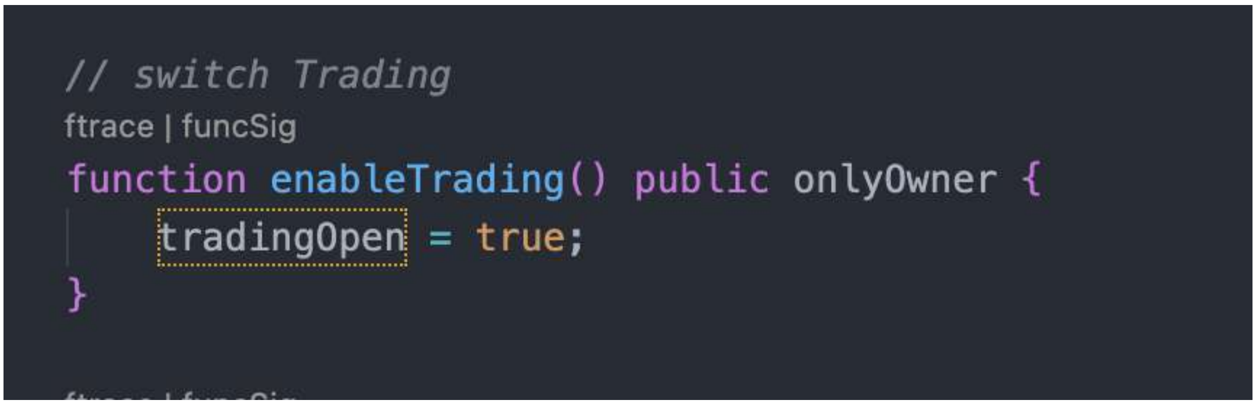
```
ftrace | funcSig
function updateSellFees(
    uint256 reward,
    uint256 liquidity,
    uint256 dev,
    uint256 strategic,
    uint256 p2e,
    uint256 stake,
    uint256 burn
) public onlyOwner {
    sellRewardFee = reward;
    sellLiquidityFee = liquidity;
    sellDevFee = dev;
    sellStrategicFee = strategic;
    sellP2EFee = p2e;
    sellStakeFee = stake;
    sellBurnFee = burn;

    sellTotalFees = reward.add(liquidity).add(dev).add(strategic);
    sellTotalFees = sellTotalFees.add(p2e).add(stake).add(burn);
    require(sellTotalFees <= 15, "Fees can not be greater than 15%");
}
```

❖ The owner can change all swap percentages



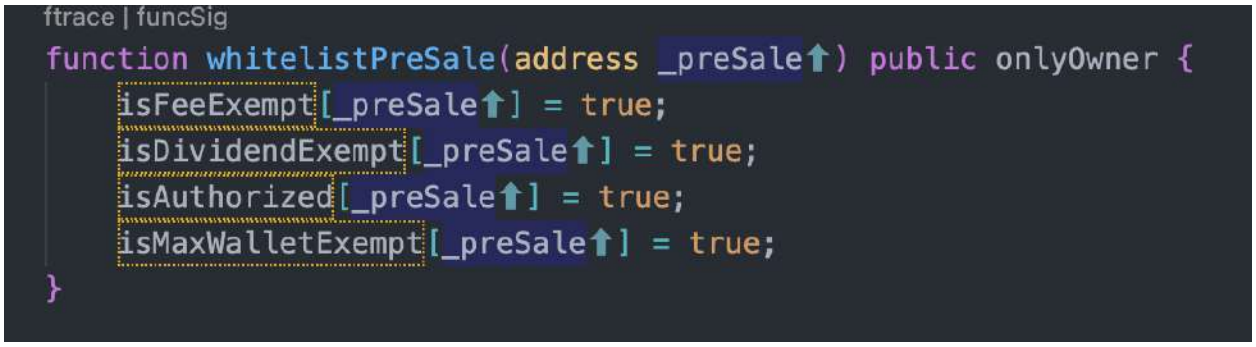
```
// update swap percentages
ftrace | funcSig
function updateSwapPercentages(
    uint256 reward,
    uint256 strategic,
    uint256 liquidity,
    uint256 dev
) public onlyOwner {
    rewardSwap = reward;
    devSwap = dev;
    liquiditySwap = liquidity;
    strategicSwap = strategic;

    totalSwap = reward.add(dev).add(liquidity).add(strategic);
}
```

❖ The owner can enable trading, once enabled cannot disable again



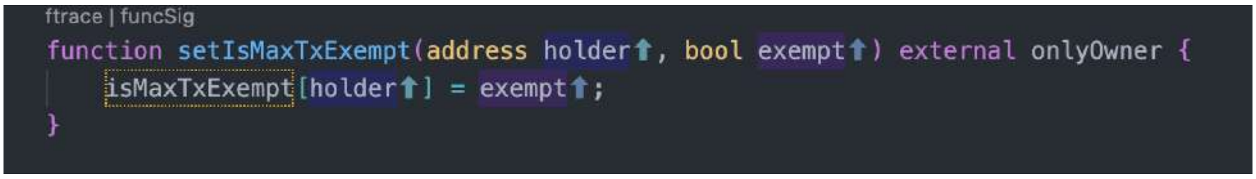
```
// switch Trading
ftrace | funcSig
function enableTrading() public onlyOwner {
    tradingOpen = true;
}
```

❖ The owner can whitelist presale address



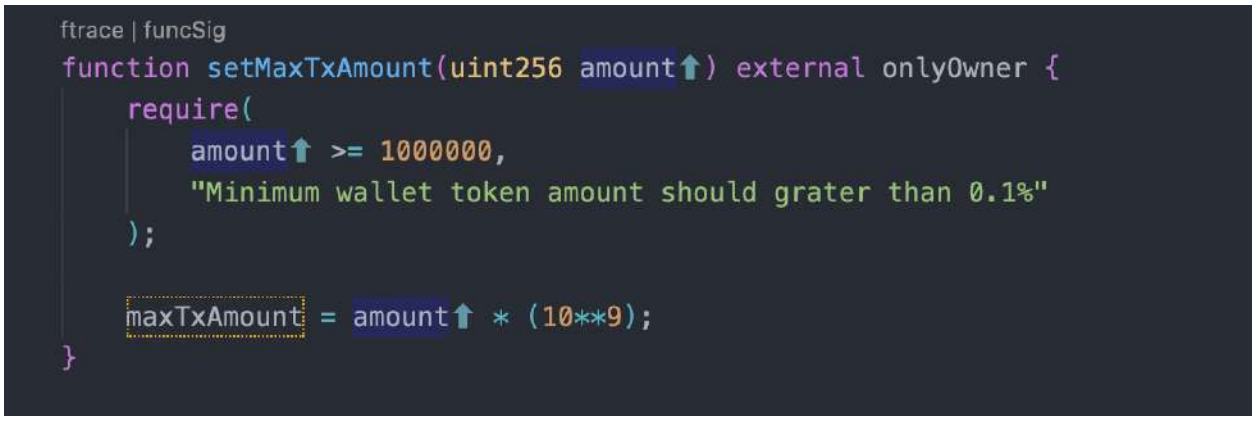
```
ftrace | funcSig
function whitelistPreSale(address _preSale) public onlyOwner {
    isFeeExempt[_preSale] = true;
    isDividendExempt[_preSale] = true;
    isAuthorized[_preSale] = true;
    isMaxWalletExempt[_preSale] = true;
}
```

❖ The owner can exclude/include wallets from max transaction



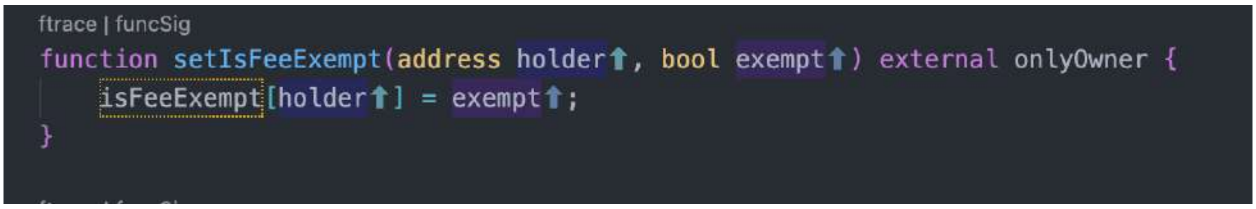
```
ftrace | funcSig
function setIsMaxTxExempt(address holder, bool exempt) external onlyOwner {
    isMaxTxExempt[holder] = exempt;
}
```

❖ The owner can change max transaction amount minimum up to 0.1%



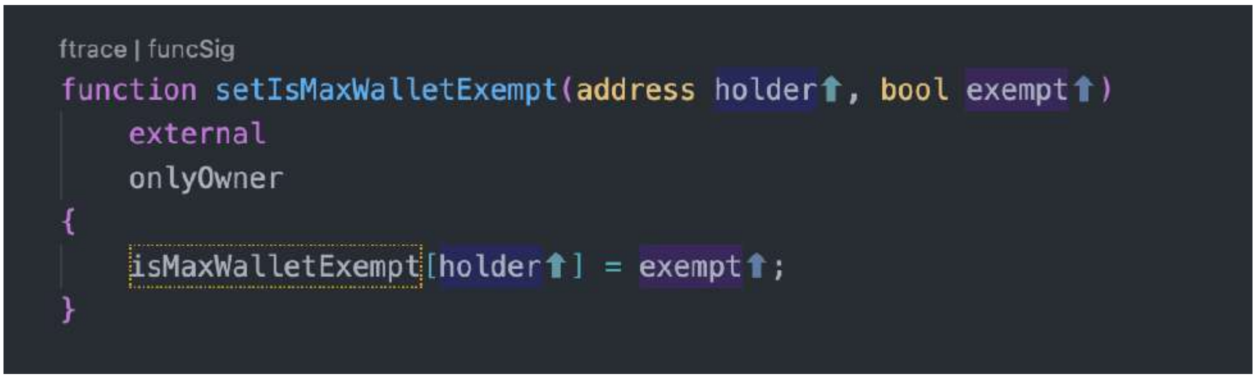
```
ftrace | funcSig
function setMaxTxAmount(uint256 amount) external onlyOwner {
    require(
        amount >= 1000000,
        "Minimum wallet token amount should grater than 0.1%"
    );

    maxTxAmount = amount * (10**9);
}
```

❖ The owner can include/exclude wallets from fees



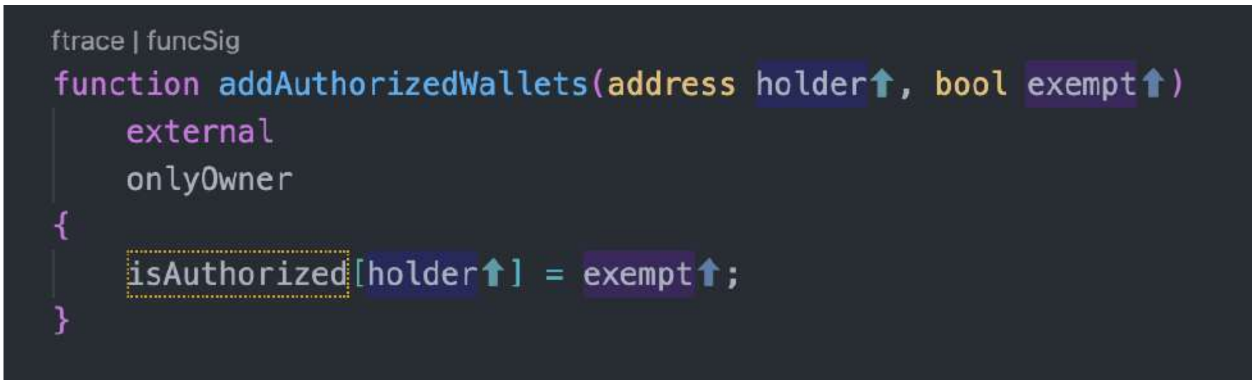
```
ftrace | funcSig
function setIsFeeExempt(address holder, bool exempt) external onlyOwner {
    isFeeExempt[holder] = exempt;
}
```

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



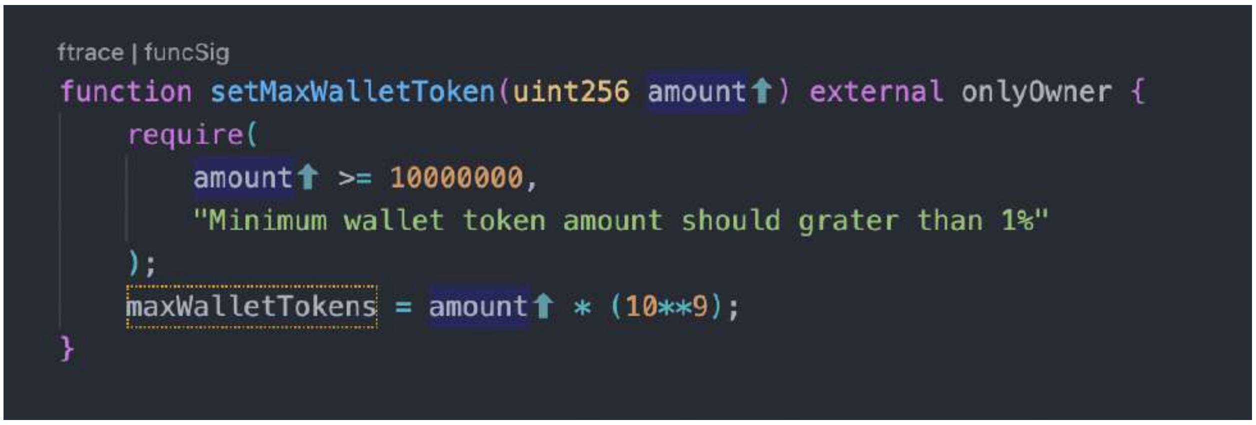
```
ftrace | funcSig
function setIsMaxWalletExempt(address holder, bool exempt)
    external
    onlyOwner
{
    isMaxWalletExempt[holder] = exempt;
}
```

❖ The owner can add/remove authorized wallets



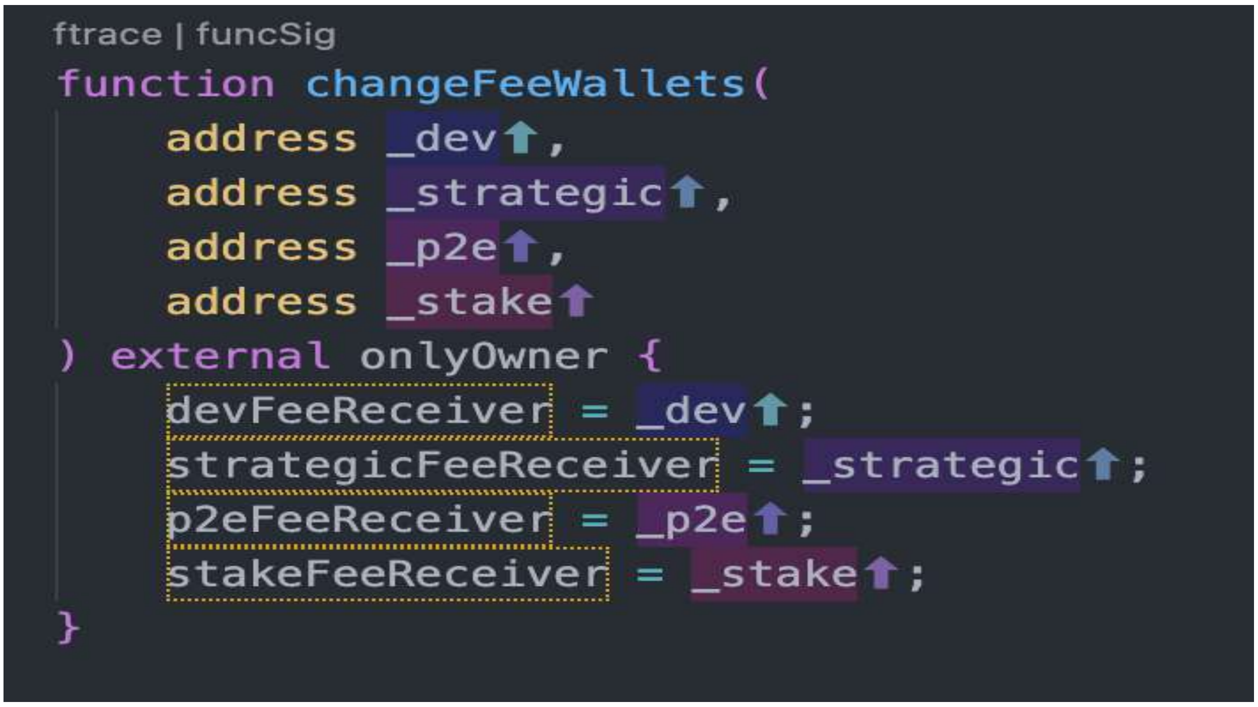
```
ftrace | funcSig
function addAuthorizedWallets(address holder, bool exempt)
    external
    onlyOwner
{
    isAuthorized[holder] = exempt;
}
```

❖ The owner can change max wallet token limit minimum up to 1%



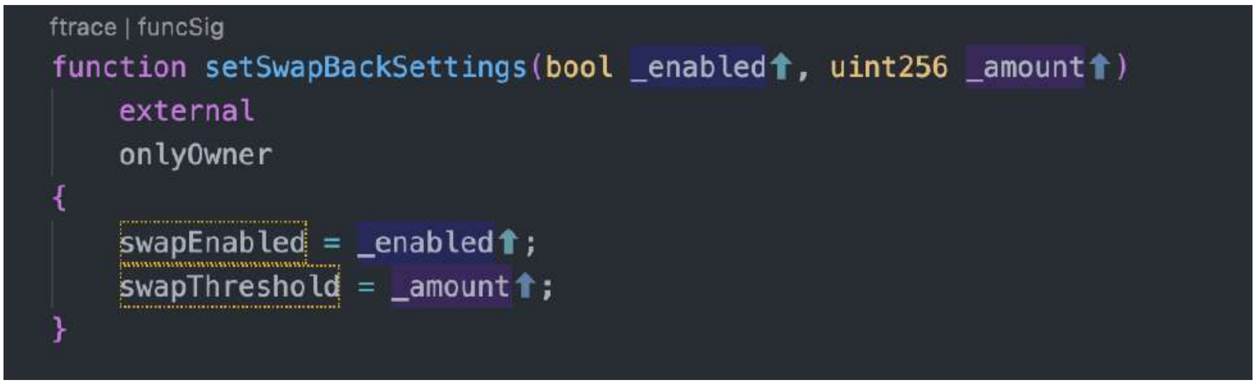
```
ftrace | funcSig
function setMaxWalletToken(uint256 amount) external onlyOwner {
    require(
        amount >= 10000000,
        "Minimum wallet token amount should grater than 1%"
    );
    maxWalletTokens = amount * (10**9);
}
```

❖ The owner can change all fee receiver wallets



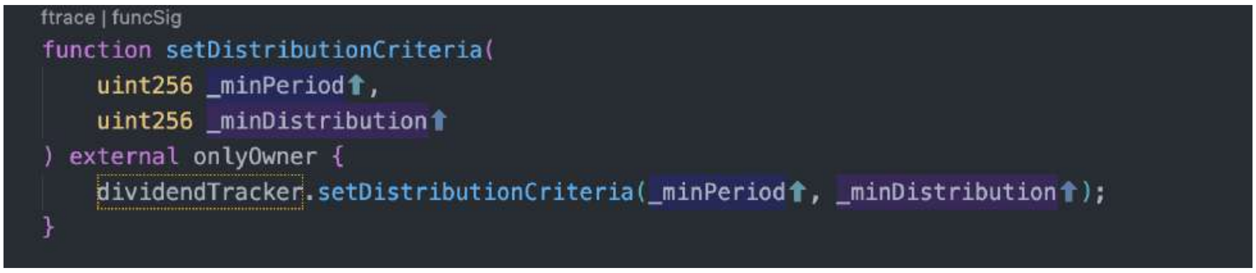
```
ftrace | funcSig
function changeFeeWallets(
    address _dev,
    address _strategic,
    address _p2e,
    address _stake
) external onlyOwner {
    devFeeReceiver = _dev;
    strategicFeeReceiver = _strategic;
    p2eFeeReceiver = _p2e;
    stakeFeeReceiver = _stake;
}
```

❖ The owner can enable/disable swapping and can change swap limit



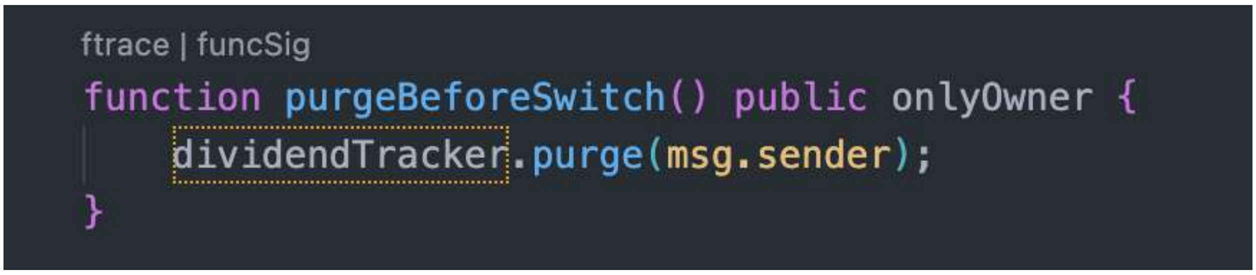
```
ftrace | funcSig
function setSwapBackSettings(bool _enabled, uint256 _amount)
    external
    onlyOwner
{
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

❖ The owner can change minimum distribution tokens and time period



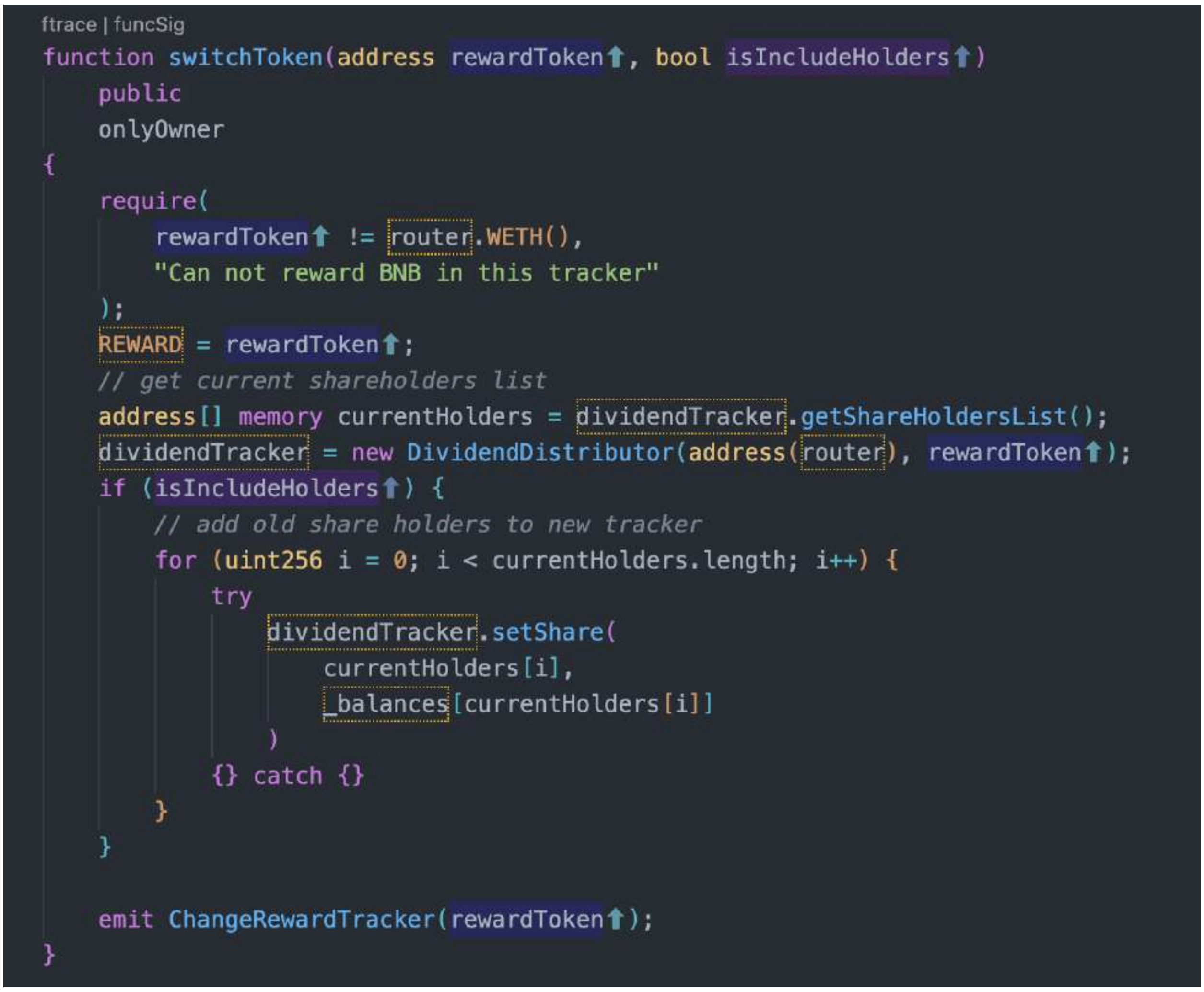
```
ftrace | funcSig
function setDistributionCriteria(
    uint256 _minPeriod,
    uint256 _minDistribution
) external onlyOwner {
    dividendTracker.setDistributionCriteria(_minPeriod, _minDistribution);
}
```

❖ The owner can get tokens balance of reward tracker before switch reward token

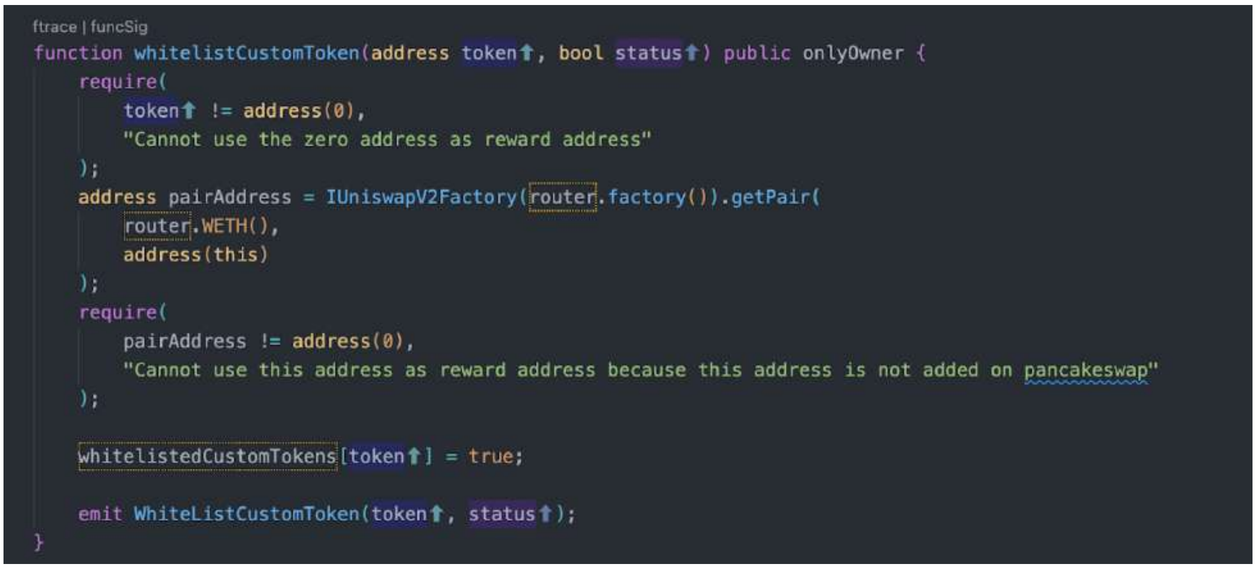


```
ftrace | funcSig
function purgeBeforeSwitch() public onlyOwner {
    dividendTracker.purge(msg.sender);
}
```

❖ The owner can change reward tracker token address


ftrace | funcSig
```
function switchToken(address rewardToken, bool isIncludeHolders)
    public
    onlyOwner
{
    require(
        rewardToken != router.WETH(),
        "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);
}
```

❖ The owner can whitelist custom tokens


ftrace | funcSig
```
function whitelistCustomToken(address token, bool status) public onlyOwner {
    require(
        token != address(0),
        "Cannot use the zero address as reward address"
    );
    address pairAddress = IUniswapV2Factory(router.factory()).getPair(
        router.WETH(),
        address(this)
    );
    require(
        pairAddress != address(0),
        "Cannot use this address as reward address because this address is not added on pancakeswap"
    );

    whitelistedCustomTokens[token] = true;

    emit WhiteListCustomToken(token, status);
}
```

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: **20**

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

Smart contract active ownership: **YES**