RugFreeCoins Audit

NUTGAIN Token

Smart Contract Security Audit

March 1, 2022

Contents

Audit details

Audited project
NUTGAIN Token

Contract Address

0xb149b030cfa47880af0bde4cd36539e4c928b3eb

Client contact

NUTGAIN Team

Blockchain

Binance smart chain

Project website

https://www.nutgain.io

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 NUTGAIN Team to perform an audit of the smart contract.

https://bscscan.com/address/0xb149b030cfa47880af0bde4cd36539e4c928b3eb#code

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

NUTGAIN is a token built on the Binance Smart Chain that is with an innovative investment use case the main purpose of which is to seek out constant revenue sources, which in turn, powers reward combined with the Staking Platform, NFT Marketplace, Defi Exchange, Crypto Wallet, and DApps. Each transaction, purchase incurs 5%, and sale incurs a 10% fee.

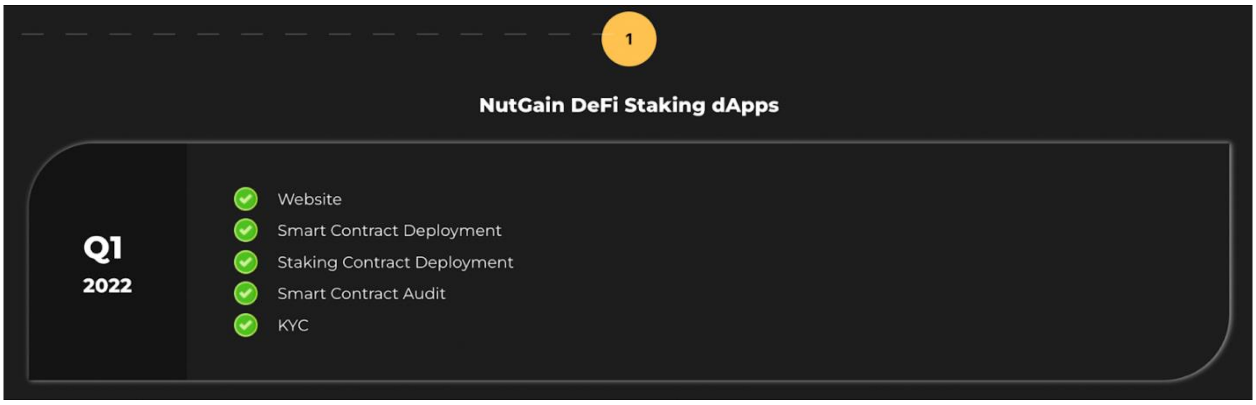
Features

- The **BUSD rewards** will be distributed among every holder proportional to how many tokens each individual holds in values of **2% when buying and 3% when selling.**

- The **sustainability fee of 2% when buying and 3% when selling for marketing and dev** is what allows NUTGAIN to hold the aforementioned promise. Tokens will be swapped into BUSD and will be sent to a marketing wallet per transaction. This way, NUTGAIN will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

- The additional component included under the sustainability section is a **liquidity fee of 2% from buying and 4% when selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

ROADMAP

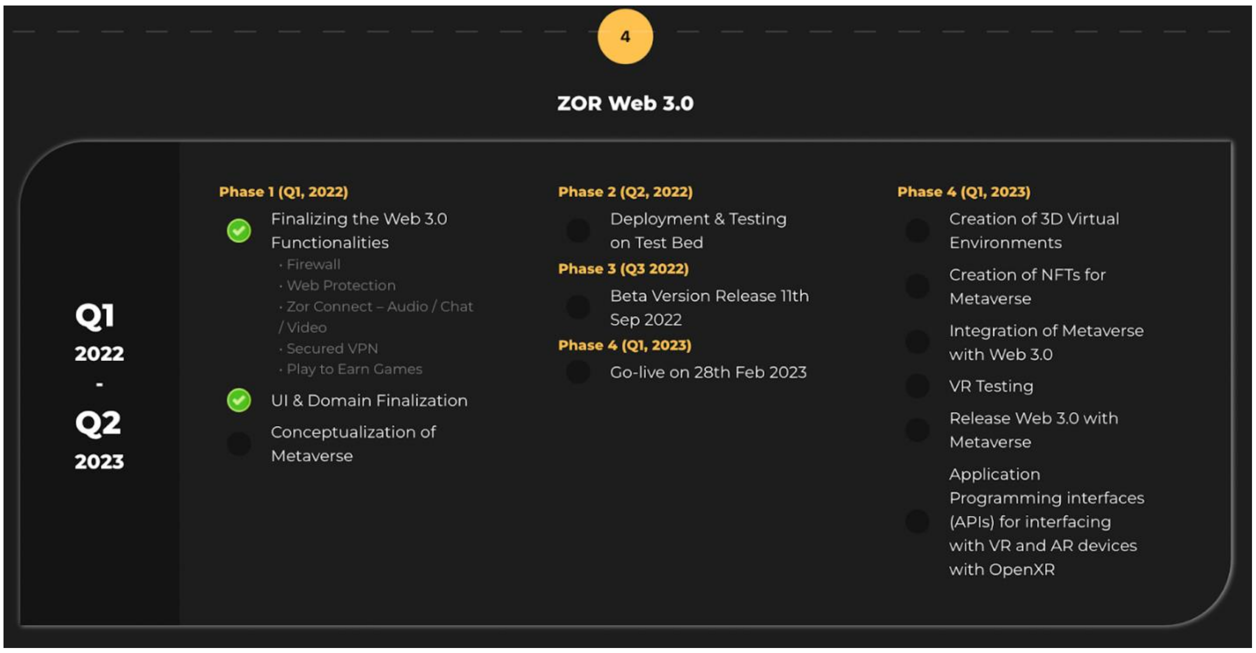


1

NutGain DeFi Staking dApps

Q1 2022

- ✓ Website
- ✓ Smart Contract Deployment
- ✓ Staking Contract Deployment
- ✓ Smart Contract Audit
- ✓ KYC



4

ZOR Web 3.0

Q1 2022 - Q2 2023

Phase 1 (Q1, 2022)

- ✓ Finalizing the Web 3.0 Functionalities
  - Firewall
  - Web Protection
  - Zor Connect – Audio / Chat / Video
  - Secured VPN
  - Play to Earn Games
- ✓ UI & Domain Finalization
- ● Conceptualization of Metaverse

Phase 2 (Q2, 2022)

- ● Deployment & Testing on Test Bed

Phase 3 (Q3 2022)

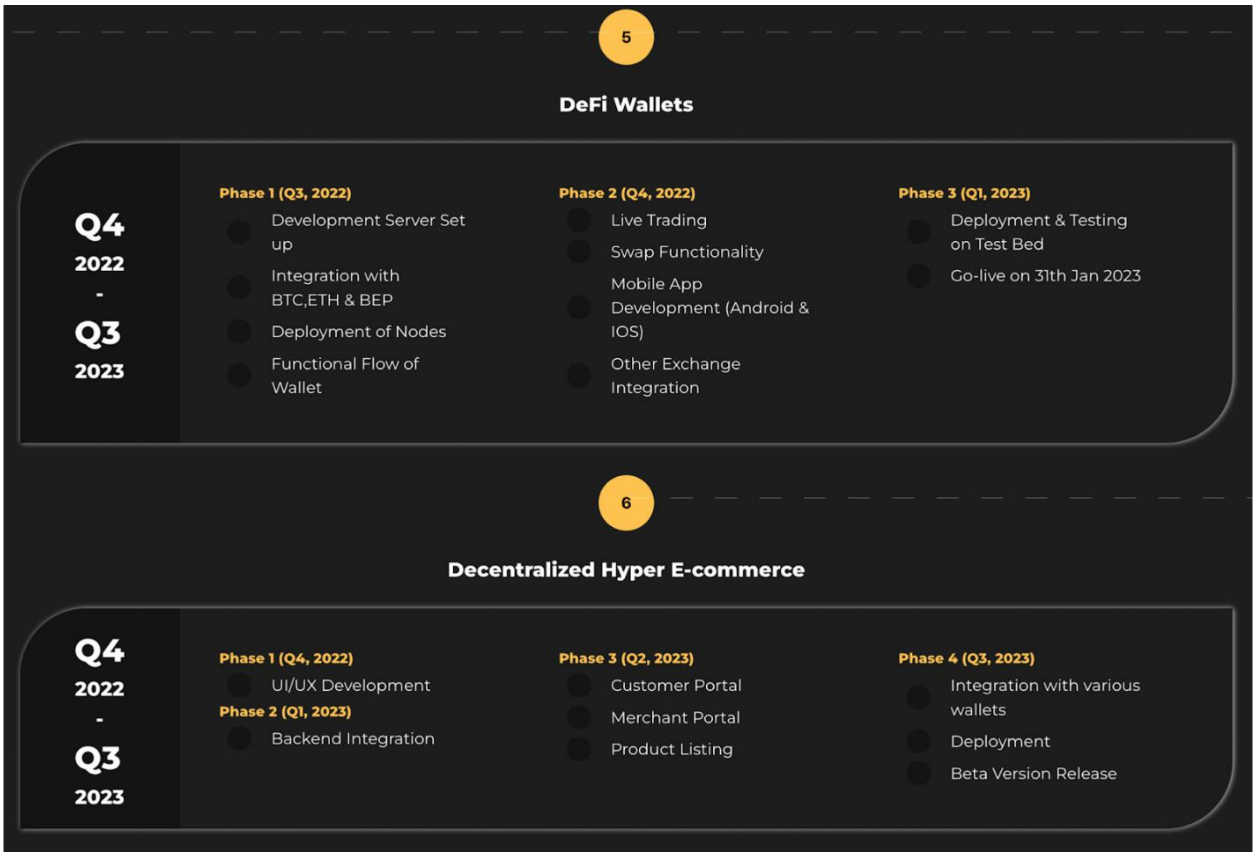
- ● Beta Version Release 11th Sep 2022

Phase 4 (Q1, 2023)

- ● Go-live on 28th Feb 2023

Phase 4 (Q1, 2023)

- ● Creation of 3D Virtual Environments
- ● Creation of NFTs for Metaverse
- ● Integration of Metaverse with Web 3.0
- ● VR Testing
- ● Release Web 3.0 with Metaverse
- ● Application Programming interfaces (APIs) for interfacing with VR and AR devices with OpenXR



5

DeFi Wallets

Q4 2022 - Q3 2023

Phase 1 (Q3, 2022)
- Development Server Set up
- Integration with BTC,ETH & BEP
- Deployment of Nodes
- Functional Flow of Wallet

Phase 2 (Q4, 2022)
- Live Trading
- Swap Functionality
- Mobile App Development (Android & IOS)
- Other Exchange Integration

Phase 3 (Q1, 2023)
- Deployment & Testing on Test Bed
- Go-live on 31th Jan 2023

6

Decentralized Hyper E-commerce

Q4 2022 - Q3 2023

Phase 1 (Q4, 2022)
- UI/UX Development

Phase 2 (Q1, 2023)
- Backend Integration

Phase 3 (Q2, 2023)
- Customer Portal
- Merchant Portal
- Product Listing

Phase 4 (Q3, 2023)
- Integration with various wallets
- Deployment
- Beta Version Release

Tokenomics

5% fee when buying

- 2% of trade goes to holders' pockets in BUSD.
- 1% of trade goes to the product development in BUSD
- 2% of trade goes to the liquidity pool.

10% fee when selling

- 3% of trade goes to holders' pockets in BUSD.
- 3% of trade goes to the product development in BUSD
- 4% of trade goes to the liquidity pool.

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 in BNB 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 collecting NFTs or trading NFTs.
- Anyone who's interested in taking part with the future plans of the NUTGAIN token.
- Anyone who's interested in making financial transactions with any other party using BNB or NUTGAIN as the currency.

Core concept

The NUTGAIN reward system

2% of each transaction when buying and 3% when selling get converted to BNB and is split amongst all holders. Holders will be eligible to receive tokens every one hour and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The **sustainability fee of 2% when buying and 3% when selling for marketing** is what allows NUTGAIN to promote the token and use funds to further the development of the platform. Tokens will be swapped into BUSD and will be sent to a marketing wallet. This way, NUTGAIN 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 when buying 2% and selling 4%, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Potential to grow with score points

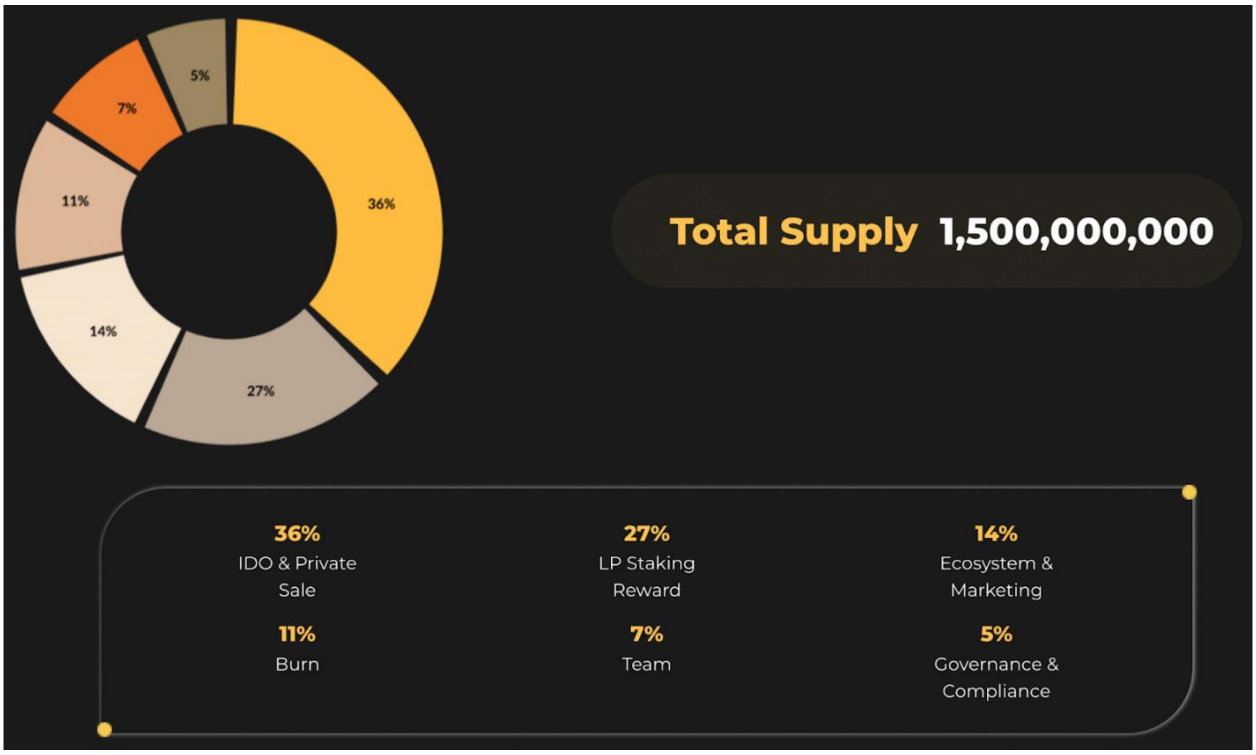
1.	Project efficiency	9/10
2.	Project uniqueness	8/10
3	Information quality	8/10
4	Service quality	9/10
5	System quality	9/10
6	Impact on the community	9/10
7	Impact on the business	9/10
8	Preparing for the future	9/10
Total Points		**8.75/10**

Contract details

Token contract details for 18th March 2022

Contract name	NUTGAIN
Contract address	0xB149B030CFA47880aF0BDE4Cd36539E4C928b3eB
Token supply	1,500,000,000
Token ticker	NUTGV2
Decimals	9
Token holders	1
Transaction count	1
Marketing wallet	0xa7cc4198d53d1415e624c88d0c790f47c2f2a115
Utility wallet	0xd510a6fd23f1d909c5cb9e20481abb344b8209d7
Contract deployer address	0x318db783dc683dD96744f5dA30c6D70Cf7Bf29fb
Contract’s current owner address	0x318db783dc683dd96744f5da30c6d70cf7bf29fb

Tokens are distributed as follows:



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**		
└	Function Name	Visibility	Mutability	Modifiers
IERC20	**Interface**			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_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❗
IRouter	**Interface**			
L	factory	External ❗		NO❗
L	WETH	External ❗		NO❗
L	addLiquidityETH	External ❗	💵	NO❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
Address	**Library**			
L	sendValue	Internal 🔒	🛑	
NUTGAIN	**Implementation**	**Context, IERC20, Ownable**		

└		Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	transfer	Public ❗	🛑	NO❗
└	isExcludedFromReward	Public ❗		NO❗
└	reflectionFromToken	Public ❗		NO❗
└	setTradingStatus	External ❗	🛑	onlyOwner
└	tokenFromReflection	Public ❗		NO❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner

└	isExcludedFromFee	Public ❗		NO❗
└	setTaxes	Public ❗	🛑	onlyOwner
└	setSellTaxes	Public ❗	🛑	onlyOwner
└	_reflectRfi	Private 🔐	🛑	
└	_takeLiquidity	Private 🔐	🛑	
└	_takeMarketing	Private 🔐	🛑	
└	_takeBurn	Private 🔐	🛑	
└	_takeDev	Private 🔐	🛑	
└	_takeUtility	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues1	Private 🔐		
└	_getRValues2	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	

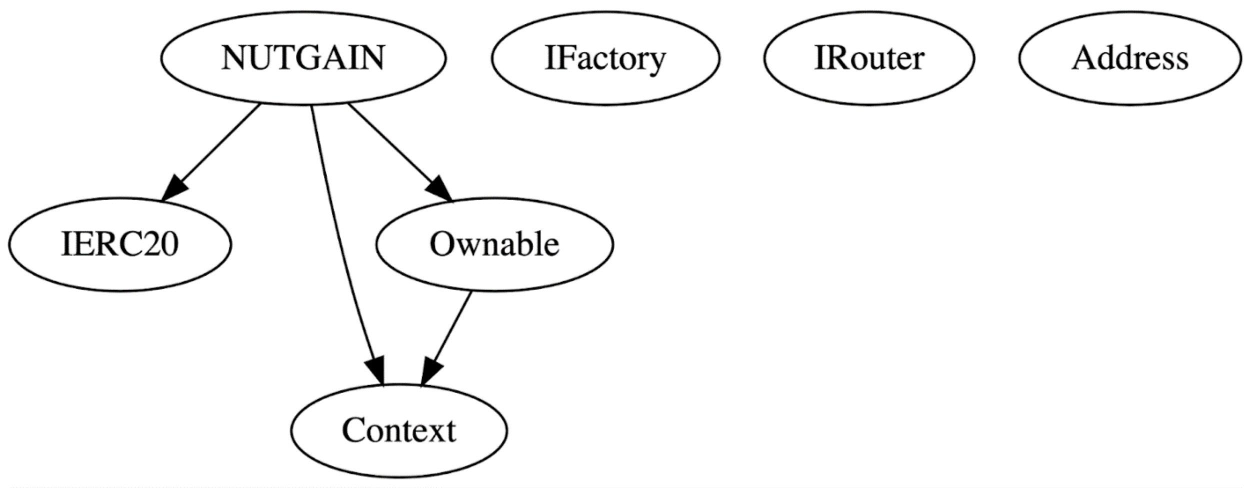
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	addLiquidity	Private 🔐	🛑	
└	swapTokensForBNB	Private 🔐	🛑	
└	bulkExcludeFee	External ❗	🛑	onlyOwner
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updateDevWallet	External ❗	🛑	onlyOwner
└	updateUtilityWallet	External ❗	🛑	onlyOwner
└	updateCooldown	External ❗	🛑	onlyOwner
└	updateSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	updateSwapEnabled	External ❗	🛑	onlyOwner
└	updateIsBlacklisted	External ❗	🛑	onlyOwner
└	bulkIsBlacklisted	External ❗	🛑	onlyOwner
└	updateAllowedTransfer	External ❗	🛑	onlyOwner
└	bulkupdateAllowedTransfer	External ❗	🛑	onlyOwner
└	updateMaxTxLimit	External ❗	🛑	onlyOwner

└	updateMaxWalletlimit	External ❗	🛑	onlyOwner
└	updateRouterAndPair	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueAnyBEP20Tokens	Public ❗	🛑	onlyOwner
└		External ❗	💵	NO❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

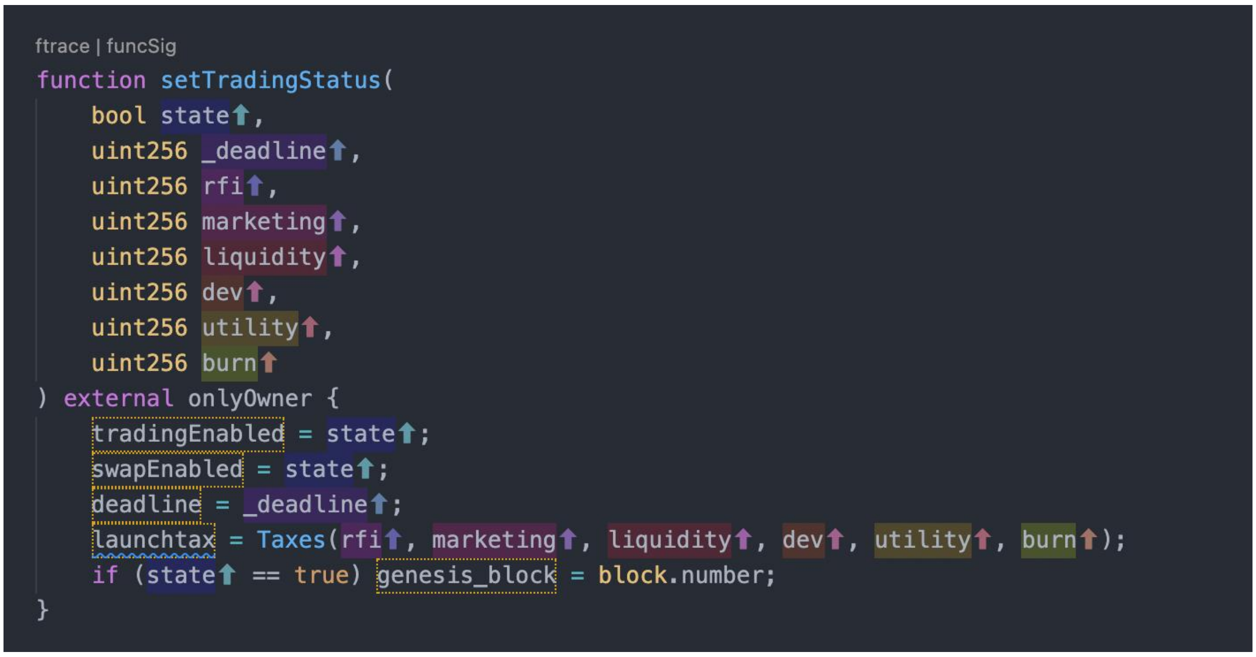
Inheritance Hierarchy



Security issue checking status

❖ **High severity issues**

The owner can enable/disable trading.



ftrace | funcSig

```
function setTradingStatus(
    bool state,
    uint256 _deadline,
    uint256 rfi,
    uint256 marketing,
    uint256 liquidity,
    uint256 dev,
    uint256 utility,
    uint256 burn
) external onlyOwner {
    tradingEnabled = state;
    swapEnabled = state;
    deadline = _deadline;
    launchtax = Taxes(rfi, marketing, liquidity, dev, utility, burn);
    if (state == true) genesis_block = block.number;
}
```

❖ **Medium severity issues**

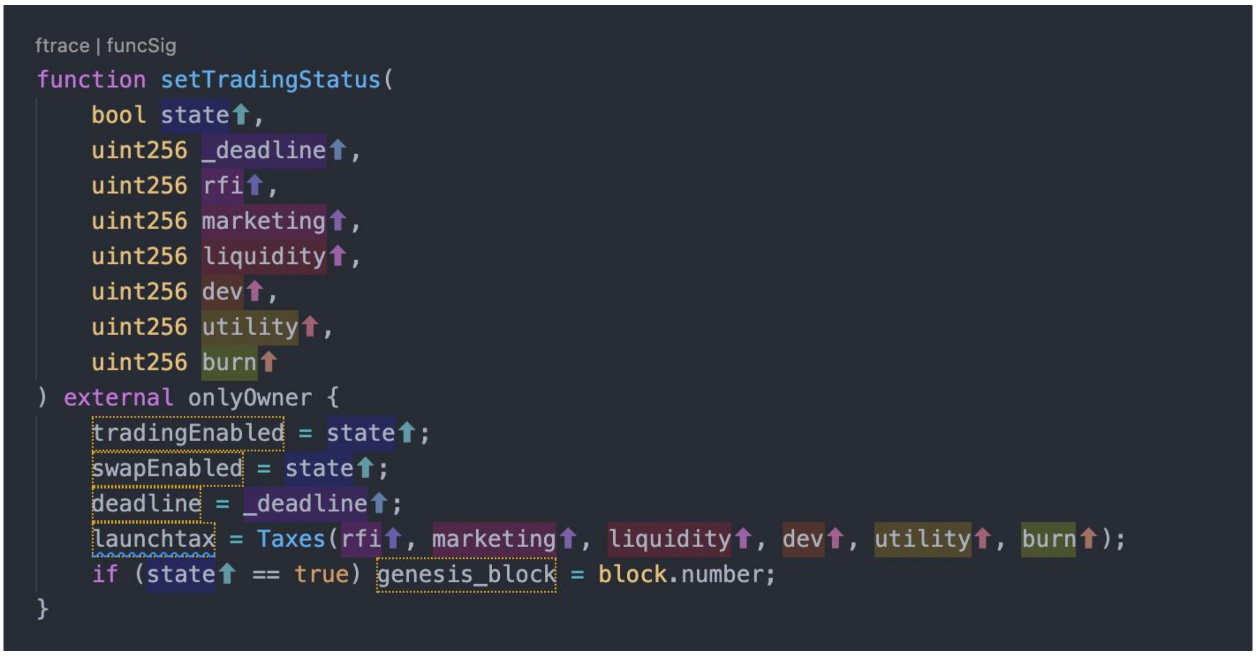
No medium severity issues found

❖ **Low severity issues**

No low severity issues found

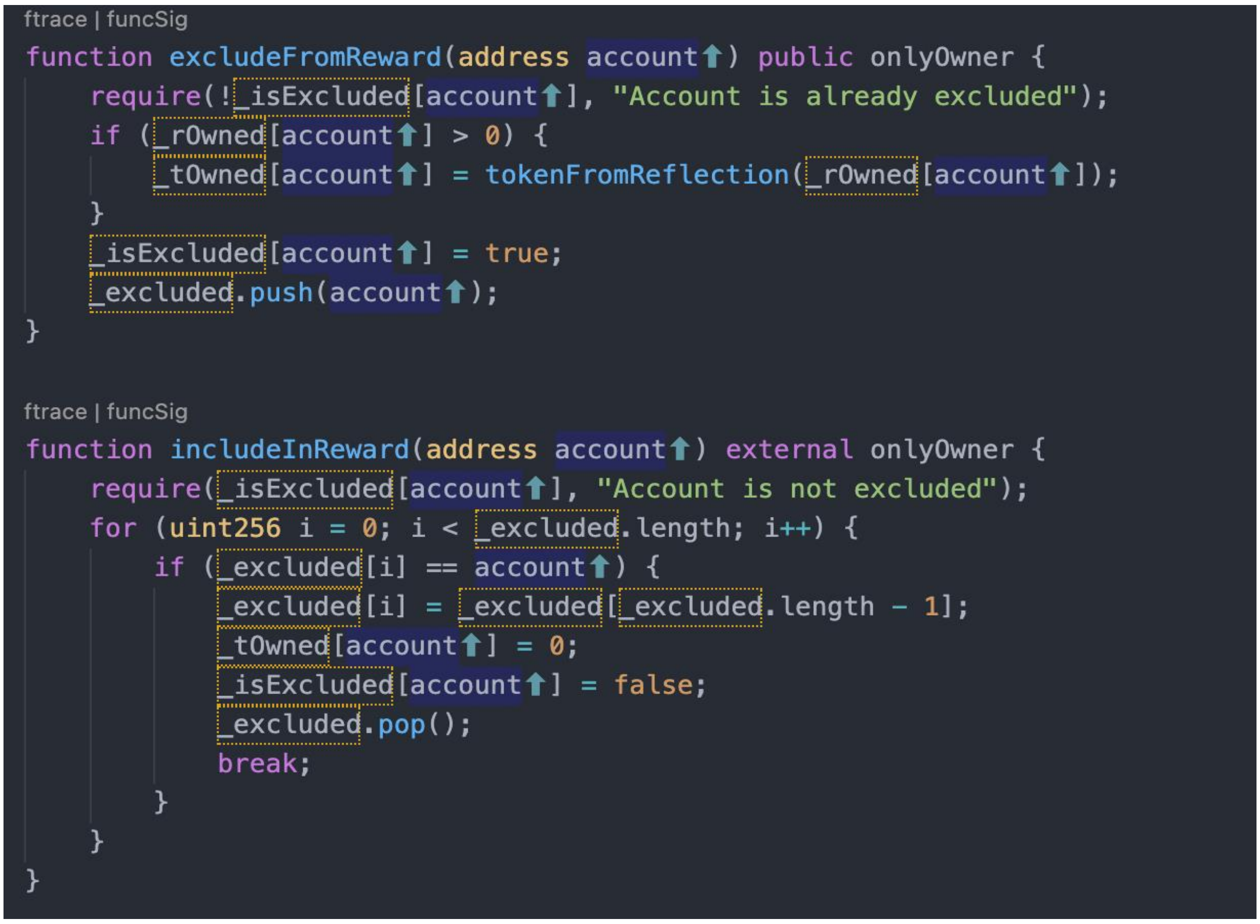
Owner privileges

❖ The owner can enable/disable trading, swapping and can change initial taxes



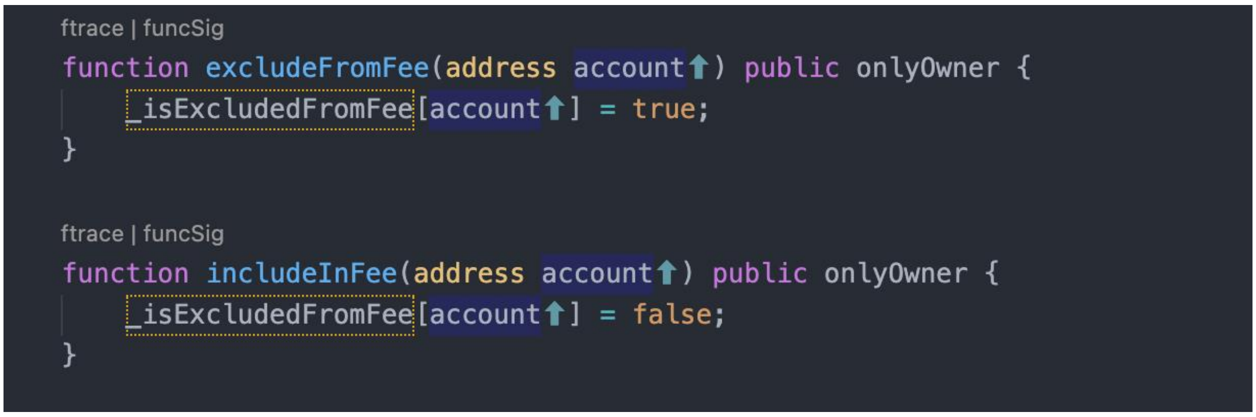
```
ftrace | funcSig
function setTradingStatus(
    bool state,
    uint256 _deadline,
    uint256 rfi,
    uint256 marketing,
    uint256 liquidity,
    uint256 dev,
    uint256 utility,
    uint256 burn
) external onlyOwner {
    tradingEnabled = state;
    swapEnabled = state;
    deadline = _deadline;
    launchtax = Taxes(rfi, marketing, liquidity, dev, utility, burn);
    if (state == true) genesis_block = block.number;
}
```

❖ The owner can include/exclude wallets from rewards



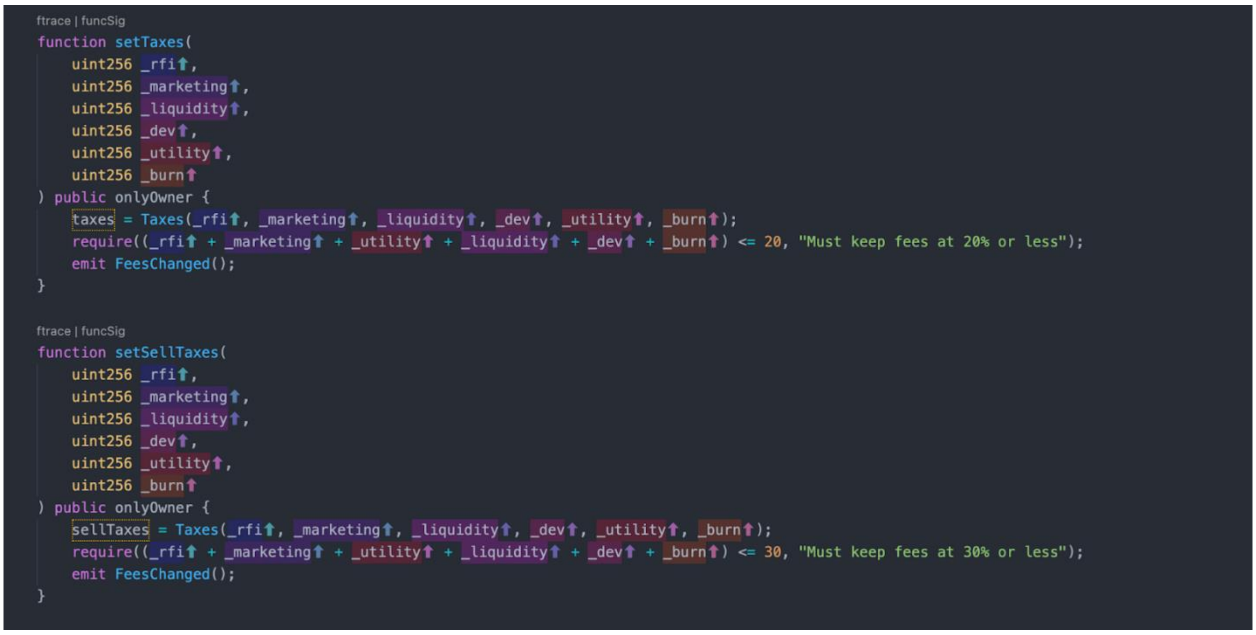
```
ftrace | funcSig
function excludeFromReward(address account) public onlyOwner {
    require(!_isExcluded[account], "Account is already excluded");
    if (_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);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

ftrace | funcSig
function includeInReward(address account) external onlyOwner {
    require(_isExcluded[account], "Account is 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 fee



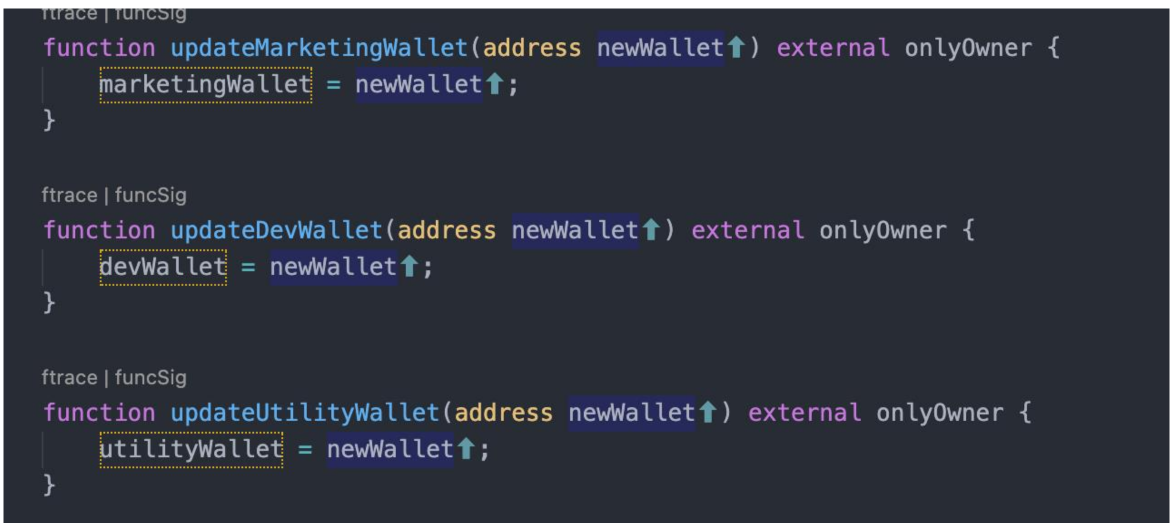
```
ftrace | funcSig
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}

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

❖ The owner can change all buy fees maximum up to 20% and sell fees maximum up to 30%



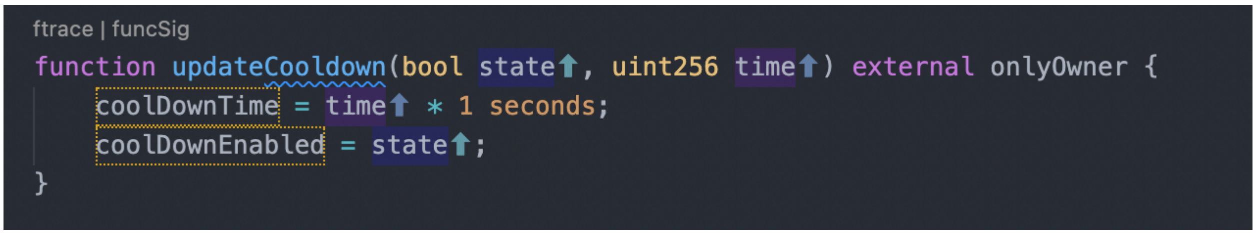
```
ftrace | funcSig
function setTaxes(
    uint256 _rfi,
    uint256 _marketing,
    uint256 _liquidity,
    uint256 _dev,
    uint256 _utility,
    uint256 _burn
) public onlyOwner {
    taxes = Taxes(_rfi, _marketing, _liquidity, _dev, _utility, _burn);
    require((_rfi + _marketing + _utility + _liquidity + _dev + _burn) <= 20, "Must keep fees at 20% or less");
    emit FeesChanged();
}

ftrace | funcSig
function setSellTaxes(
    uint256 _rfi,
    uint256 _marketing,
    uint256 _liquidity,
    uint256 _dev,
    uint256 _utility,
    uint256 _burn
) public onlyOwner {
    sellTaxes = Taxes(_rfi, _marketing, _liquidity, _dev, _utility, _burn);
    require((_rfi + _marketing + _utility + _liquidity + _dev + _burn) <= 30, "Must keep fees at 30% or less");
    emit FeesChanged();
}
```

❖ The owner can change marketing, dev and utility wallets



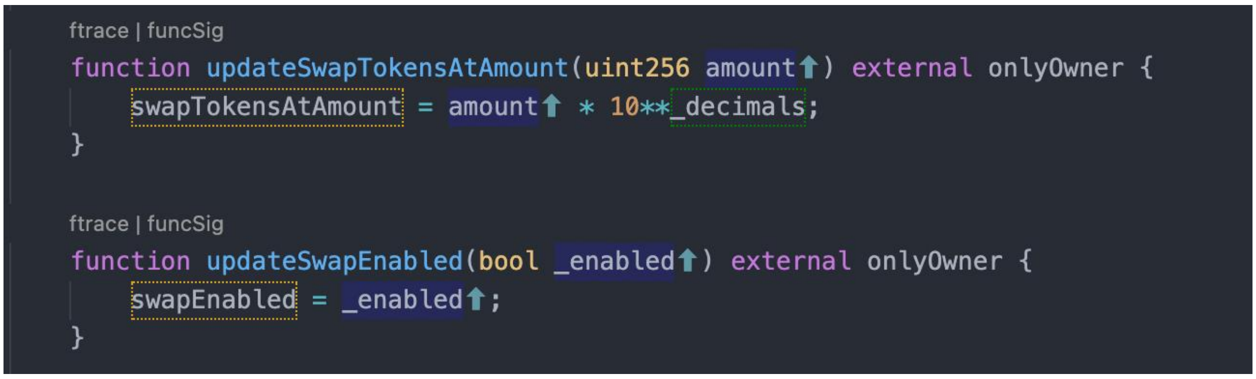
```
ftrace | funcSig
function updateMarketingWallet(address newWallet) external onlyOwner {
    marketingWallet = newWallet;
}

ftrace | funcSig
function updateDevWallet(address newWallet) external onlyOwner {
    devWallet = newWallet;
}

ftrace | funcSig
function updateUtilityWallet(address newWallet) external onlyOwner {
    utilityWallet = newWallet;
}
```

❖ The owner can enable/disable sell cool down and can change sell cool downtime



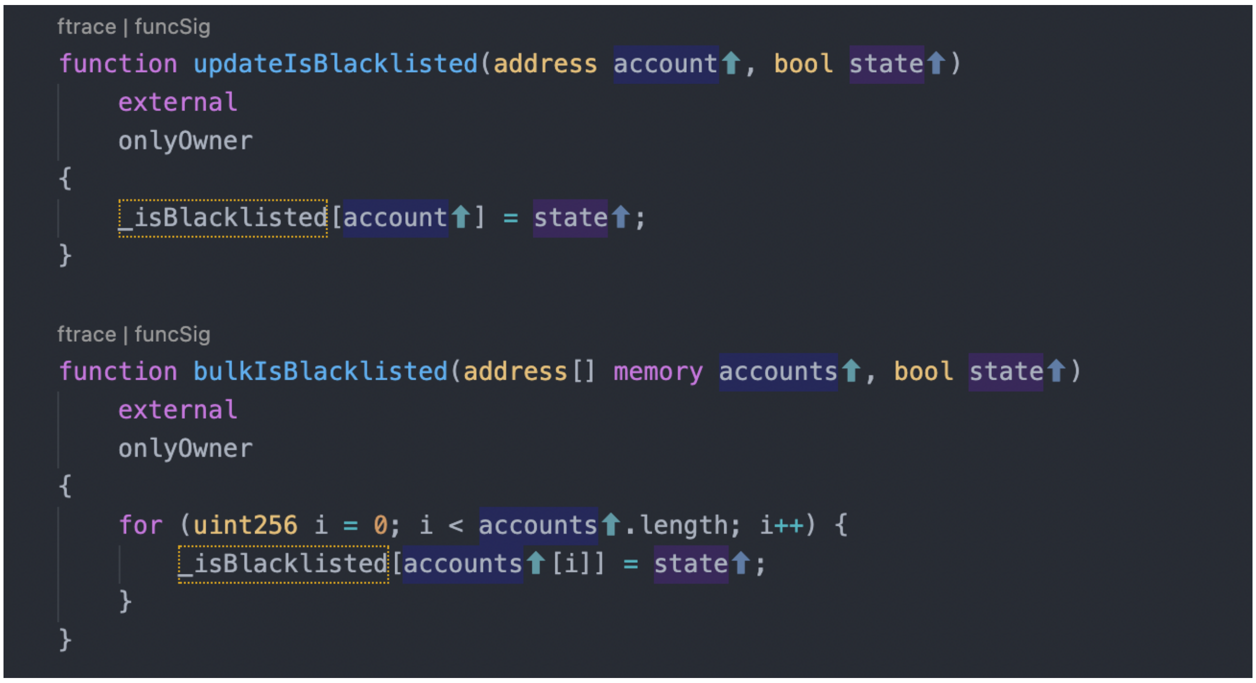
```
ftrace | funcSig
function updateCooldown(bool state, uint256 time) external onlyOwner {
    coolDownTime = time * 1 seconds;
    coolDownEnabled = state;
}
```

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



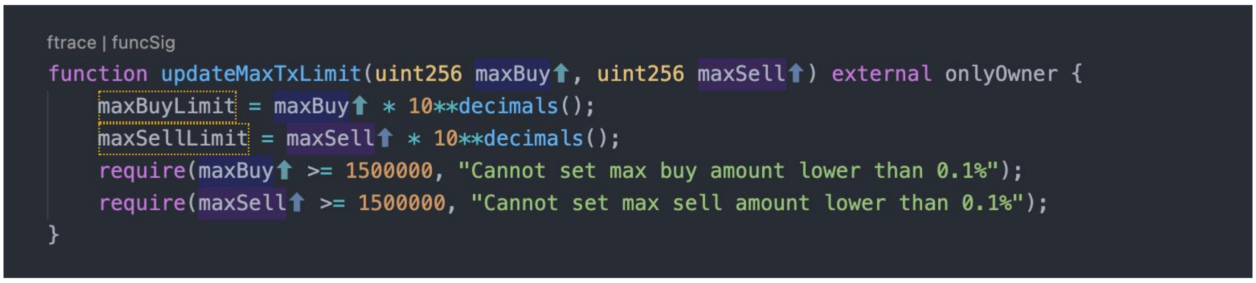
```
ftrace | funcSig
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    swapTokensAtAmount = amount * 10**_decimals;
}

ftrace | funcSig
function updateSwapEnabled(bool _enabled) external onlyOwner {
    swapEnabled = _enabled;
}
```

❖ The owner can add/remove wallets from blacklist



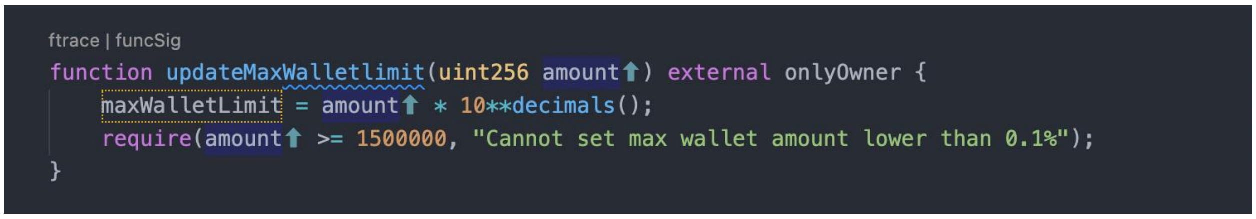
```
ftrace | funcSig
function updateIsBlacklisted(address account, bool state)
    external
    onlyOwner
{
    _isBlacklisted[account] = state;
}

ftrace | funcSig
function bulkIsBlacklisted(address[] memory accounts, bool state)
    external
    onlyOwner
{
    for (uint256 i = 0; i < accounts.length; i++) {
        _isBlacklisted[accounts[i]] = state;
    }
}
```

❖ The owner can change max buy and sell limit



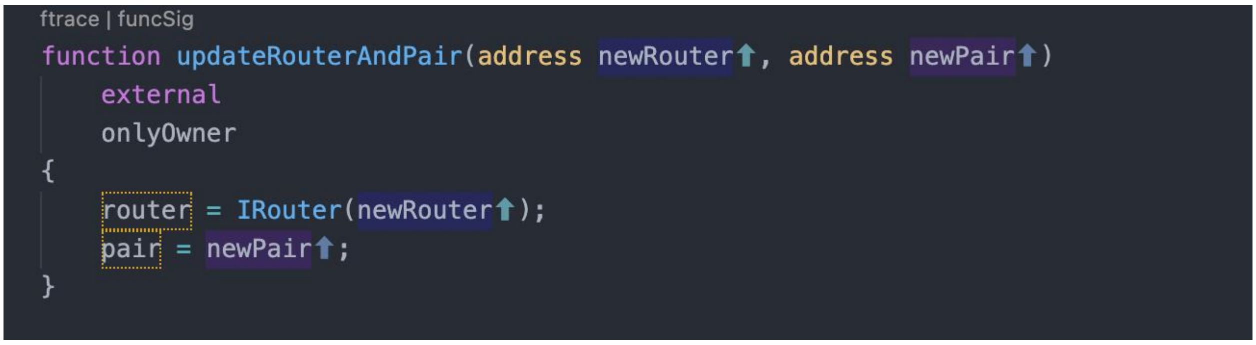
```
ftrace | funcSig
function updateMaxTxLimit(uint256 maxBuy, uint256 maxSell) external onlyOwner {
    maxBuyLimit = maxBuy * 10**decimals();
    maxSellLimit = maxSell * 10**decimals();
    require(maxBuy >= 1500000, "Cannot set max buy amount lower than 0.1%");
    require(maxSell >= 1500000, "Cannot set max sell amount lower than 0.1%");
}
```

❖ The owner can change max wallet token amount



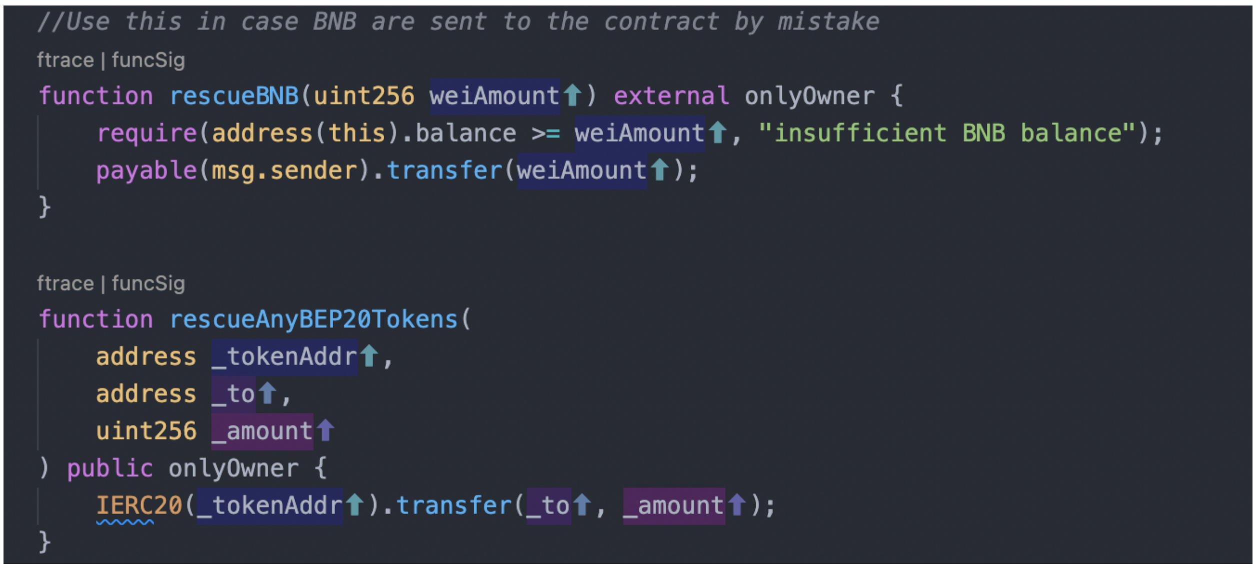
```
ftrace | funcSig
function updateMaxWalletlimit(uint256 amount) external onlyOwner {
    maxWalletLimit = amount * 10**decimals();
    require(amount >= 1500000, "Cannot set max wallet amount lower than 0.1%");
}
```

❖ The owner can change router and pair address



```
ftrace | funcSig
function updateRouterAndPair(address newRouter, address newPair)
    external
    onlyOwner
{
    router = IRouter(newRouter);
    pair = newPair;
}
```

❖ The owner can get bnb and bep20 tokens in 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);
}

ftrace | funcSig
function rescueAnyBEP20Tokens(
    address _tokenAddr,
    address _to,
    uint256 _amount
) public onlyOwner {
    IERC20(_tokenAddr).transfer(_to, _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: **PASSED**

Number of risk issues: **1**

Solidity code functional issue level: **PASSED**

Number of owner privileges: **12**

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

Smart contract active ownership: **YES**