RUGFREECOINS

ZHL Token

RugfreeCoins Verified on August 24th, 2023

Overview

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

❌ The owner can set a max transaction limit

✅ The owner can’t pause trading once it’s enabled

❌ The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

❌ The owner can change fees over 20%.

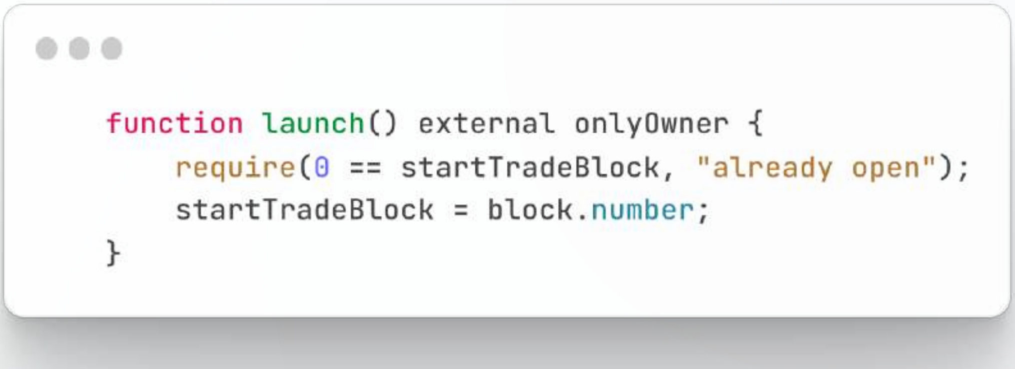
❌ The owner can't blacklist wallets.

❌ The owner can’t set a maximum wallet limit

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

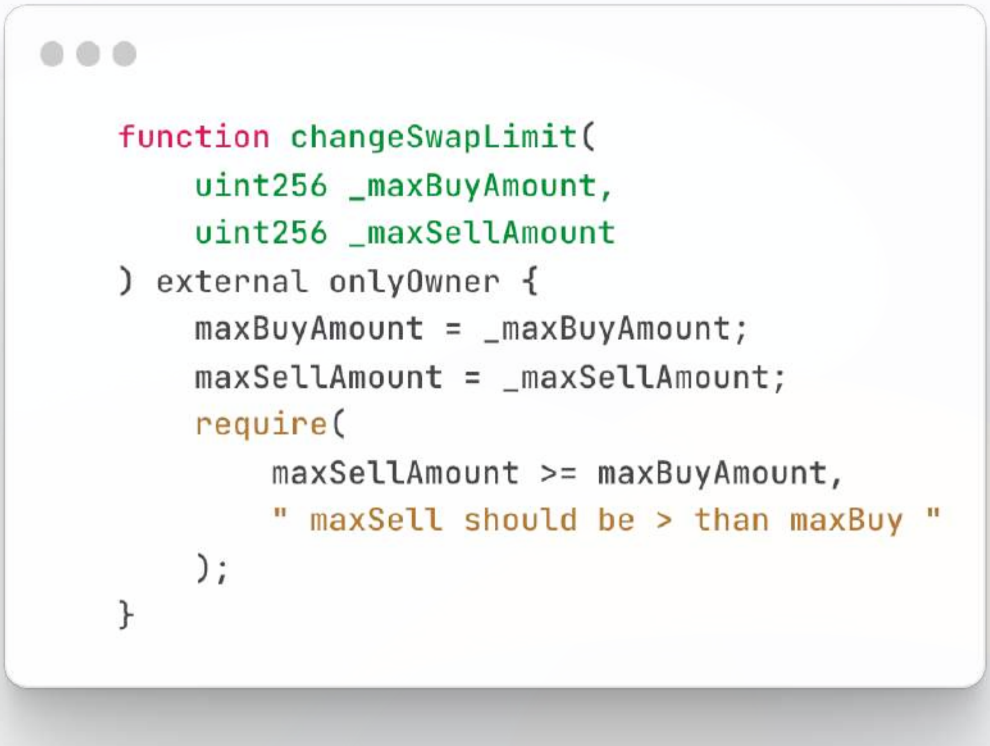
- **High severity issues**

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.



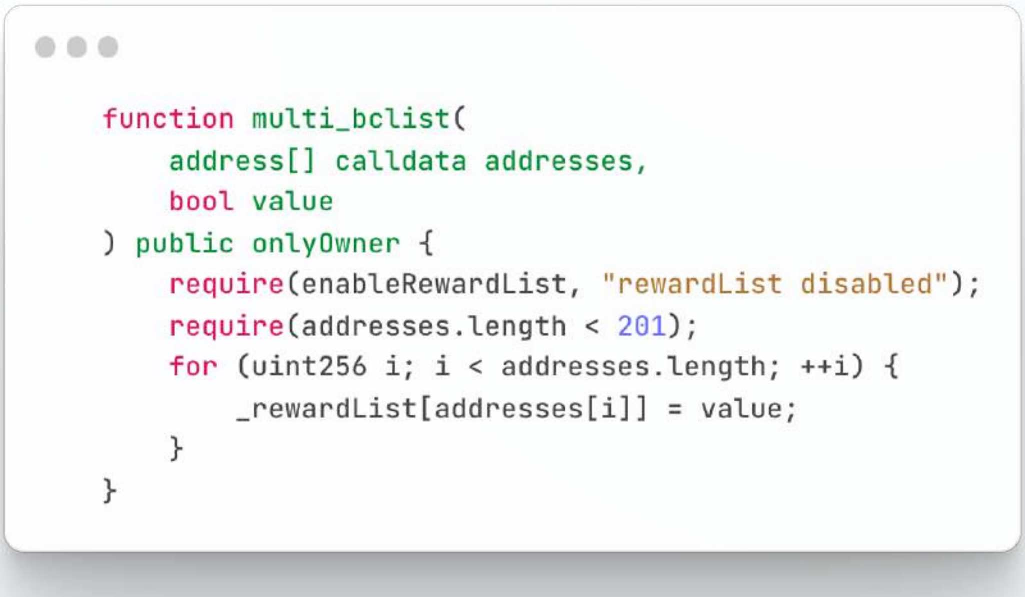
```
function launch() external onlyOwner {
    require(0 == startTradeBlock, "already open");
    startTradeBlock = block.number;
}
```

The owner can change the max buy and sell limit up to 0, the owner can stop trading by changing this to a very low amount



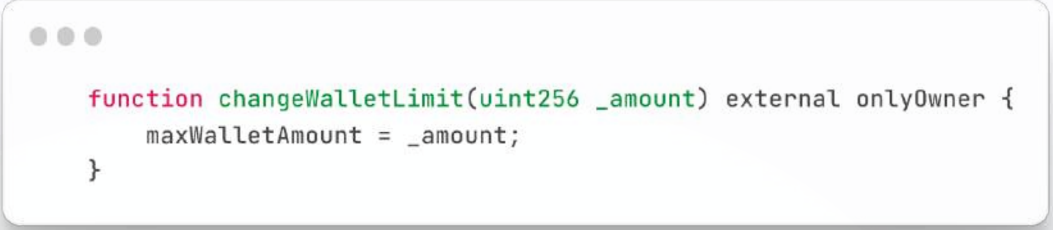
```
function changeSwapLimit(
    uint256 _maxBuyAmount,
    uint256 _maxSellAmount
) external onlyOwner {
    maxBuyAmount = _maxBuyAmount;
    maxSellAmount = _maxSellAmount;
    require(
        maxSellAmount >= maxBuyAmount,
        " maxSell should be > than maxBuy "
    );
}
```

The owner can block wallets from the contract



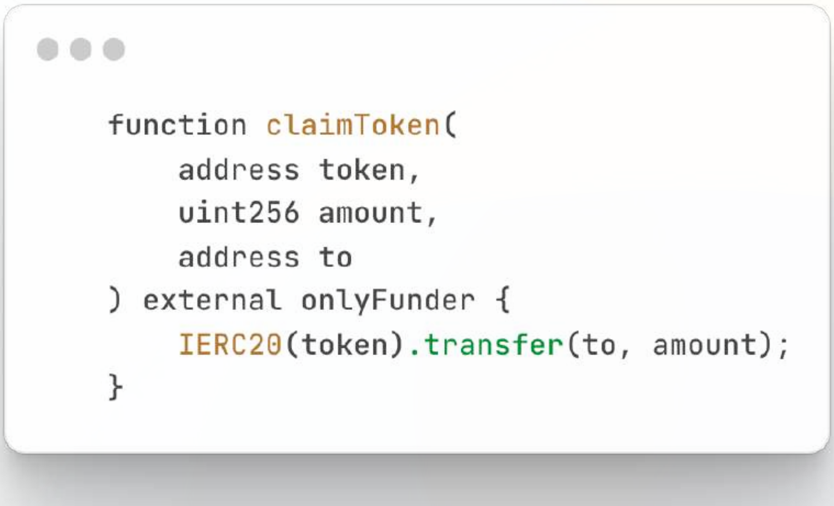
```
function multi_bclist(
    address[] calldata addresses,
    bool value
) public onlyOwner {
    require(enableRewardList, "rewardList disabled");
    require(addresses.length < 201);
    for (uint256 i; i < addresses.length; ++i) {
        _rewardList[addresses[i]] = value;
    }
}
```

The owner can set the wallet limit to 0



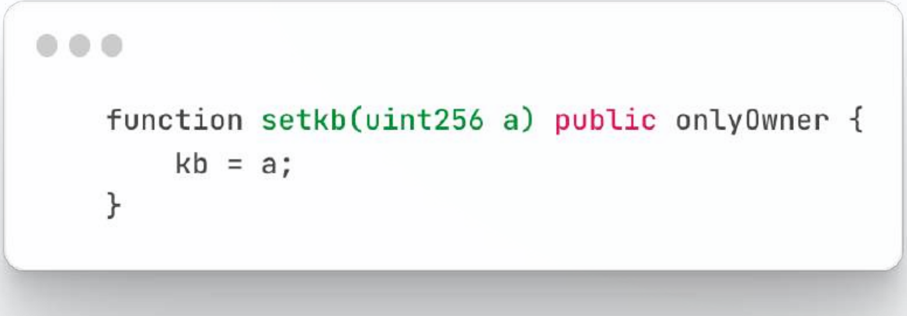
```
function changeWalletLimit(uint256 _amount) external onlyOwner {
    maxWalletAmount = _amount;
}
```

The owner can claim native tokens from the contract



```
function claimToken(
    address token,
    uint256 amount,
    address to
) external onlyFunder {
    IERC20(token).transfer(to, amount);
}
```

The owner can change the number of killer blocks without any limit



```
function setkb(uint256 a) public onlyOwner {
    kb = a;
}
```

Contents

Audit details

Audited project

ZHL Token

Contract Address

0x5b7f893434471128d1EF72f8F536C20986FBa67A

Client contact

ZHL Token Team

Blockchain

Binance Smart chain

Project website

http://www.zhlbsc.top

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 **ZHL Token Team** to perform an audit of the smart contract.

https://bscscan.com/address/0x5b7f893434471128d1ef72f8f536c20986fba67a

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.

Tokenomics

▲ 3.8% tax when buying & selling (24/08/2023)

0.8% of trade goes to the Fund fee wallet in BNB
3% of trade is distributed among holders as rewards in BNB.
0% of trade goes to the Liquidity Pool.

Target market and the concept

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

Potential to grow with score points

⚡ Project efficiency	8 / 10
🌟 Project uniqueness	7 / 10
📊 Information quality	8 / 10
👌 Service quality	8 / 10
💻 System quality	8 / 10
🌍 Impact on the community	8 / 10
💼 Impact on the business	9 / 10
🔮 Preparing for the future	8 / 10
🔒 Smart contract security	5 / 10
🛠 Smart contract functionality assessment	9/ 10
🏆 **Total Score**	**7.8**/ 10

Contract details

Token contract details for 24th of August 2023

Contract name	**ZHL (ZHL)**
Contract address	**0x5b7f893434471128d1EF72f8F536C20986FBa67A**
Token supply	**1,000,000**
Token ticker	**ZHL**
Decimals	**18**
Token holders	**2**
Transaction count	**3**
Contract deployer address	**0xF8FfD616094a5E0f920a06b4b5a977a316E42937**
Contract's current owner address	**0xF8FfD616094a5E0f920a06b4b5a977a316E42937**
Reward Token Distributor	**0x152F8B77aDbb5620f906D0c2a14f0516Fac8Fe74**

Contract code function details

№	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 & centralisation	Access control of owners	HIGH
		Business logics	PASS
		Business implementation	PASS
4	Integer overflow/underflow		PASS
5	Reentrancy		PASS
6	Exceptional reachable state		PASS
7	Transaction ordering dependence		PASS
8	Block properties dependence		PASS
9	Pseudo random number generator (PRNG)		PASS
10	DoS (Denial of Service)		PASS
11	Token vesting implementation		PASS
12	Fake deposit		PASS
13	Event security		PASS

Contract description table

The below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions, and implementations with their visibility and mutability.

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
IERC20	**Interface**			
L	decimals	External ❗		NO ❗
L	symbol	External ❗		NO ❗
L	name	External ❗		NO ❗
L	totalSupply	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	transfer	External ❗	🛑	NO ❗
L	allowance	External ❗		NO ❗
L	approve	External ❗	🛑	NO ❗
L	transferFrom	External ❗	🛑	NO ❗
ISwapRouter	**Interface**			
L	factory	External ❗		NO ❗
L	WETH	External ❗		NO ❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗

L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
L	addLiquidity	External ❗	🛑	NO ❗
L	addLiquidityETH	External ❗	💵	NO ❗
ISwapFactory	**Interface**			
L	createPair	External ❗	🛑	NO ❗
L	getPair	External ❗		NO ❗
Ownable	**Implementation**			
L		Public ❗	🛑	NO ❗
L	owner	Public ❗		NO ❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
Token Distributor	**Implementation**			
L		Public ❗	🛑	NO ❗
ISwapPair	**Interface**			
L	getReserves	External ❗		NO ❗
L	token0	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	totalSupply	External ❗		NO ❗
FatToken	**Implementation**	**IERC20, Ownable**		

L		Public ❗	🛑	NO ❗
L	symbol	External ❗		NO ❗
L	name	External ❗		NO ❗
L	decimals	External ❗		NO ❗
L	totalSupply	Public ❗		NO ❗
L	balanceOf	Public ❗		NO ❗
L	transfer	Public ❗	🛑	NO ❗
L	allowance	Public ❗		NO ❗
L	approve	Public ❗	🛑	NO ❗
L	transferFrom	Public ❗	🛑	NO ❗
L	_approve	Private 🔐	🛑	
L	setisMaxEatExempt	External ❗	🛑	onlyOwner
L	setkb	Public ❗	🛑	onlyOwner
L	isReward	Public ❗		NO ❗
L	setAirDropEnable	Public ❗	🛑	onlyOwner
L	_basicTransfer	Internal 🔒	🛑	
L	setAirdropNumbs	Public ❗	🛑	onlyOwner
L	setEnableTransferFee	Public ❗	🛑	onlyOwner
L	_isAddLiquidity	Internal 🔒		
L	_isRemoveLiquidity	Internal 🔒		
L	_transfer	Private 🔐	🛑	
L	_funTransfer	Private 🔐	🛑	
L	setTransferFee	Public ❗	🛑	onlyOwner
L	setAddLiquidityFee	Public ❗	🛑	onlyOwner

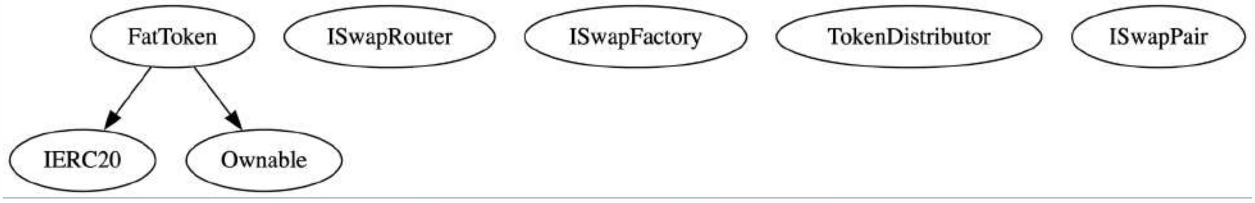
L	setRemoveLiquidityFee	Public ❗	🛑	onlyOwner
L	_tokenTransfer	Private 🔐	🛑	
L	swapTokenForFund	Private 🔐	🛑	lockTheSwap
L	_takeTransfer	Private 🔐	🛑	
L	setFundAddress	External ❗	🛑	onlyOwner
L	isContract	Private 🔐		
L	startLP	External ❗	🛑	onlyOwner
L	stopLP	External ❗	🛑	onlyOwner
L	launch	External ❗	🛑	onlyOwner
L	setFeeWhiteList	Public ❗	🛑	onlyOwner
L	completeCustoms	External ❗	🛑	onlyOwner
L	multi_bclist	Public ❗	🛑	onlyOwner
L	disableKillBatchBot	Public ❗	🛑	onlyOwner
L	disableSwapLimit	Public ❗	🛑	onlyOwner
L	disableWalletLimit	Public ❗	🛑	onlyOwner
L	disableChangeTax	Public ❗	🛑	onlyOwner
L	setSwapPairList	External ❗	🛑	onlyOwner
L	changeSwapLimit	External ❗	🛑	onlyOwner
L	changeWalletLimit	External ❗	🛑	onlyOwner
L	claimBalance	External ❗	🛑	NO ❗
L	claimToken	External ❗	🛑	onlyFunder
L		External ❗	💵	NO ❗

L	addHolder	Private 🔐	🛑	
L	setProcessRewardWaitBlock	Public ❗	🛑	onlyOwner
L	processReward	Private 🔐	🛑	
L	setHolderRewardCondition	External ❗	🛑	onlyOwner
L	setExcludeHolder	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

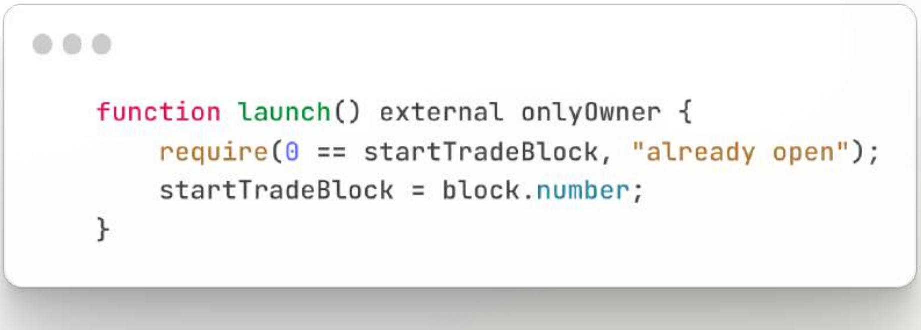
Inheritance Hierarchy



Security issue checking status

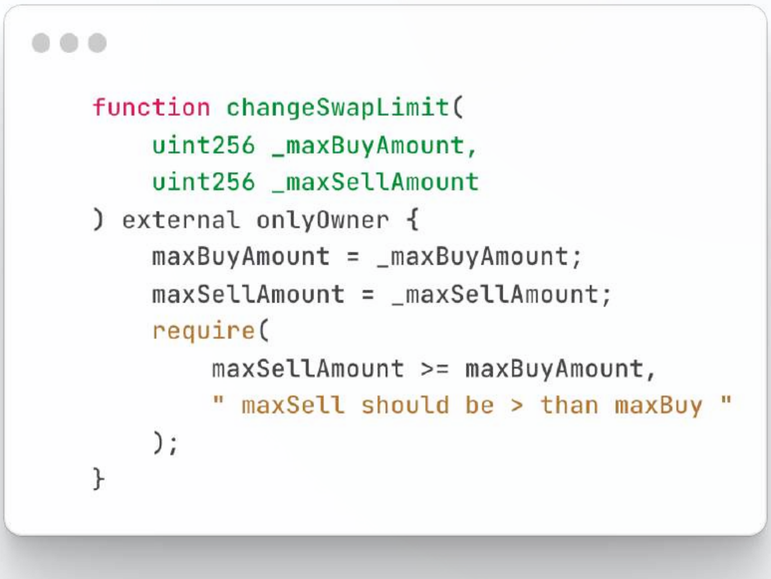
❖ High severity issues

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.



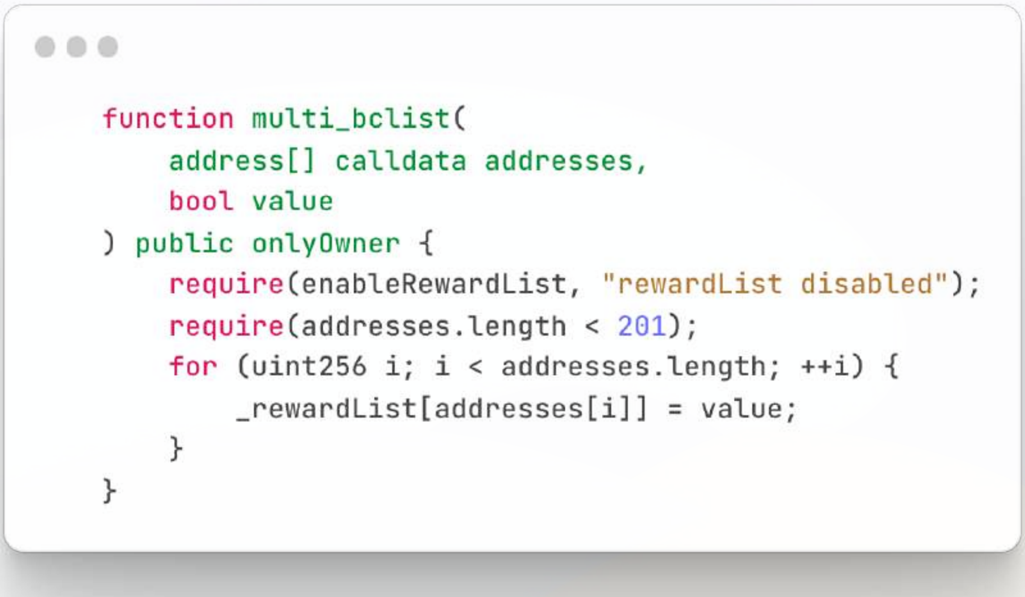
```
function launch() external onlyOwner {
    require(0 == startTradeBlock, "already open");
    startTradeBlock = block.number;
}
```

Owner can change the max buy and sell limit up to 0, the owner can stop trading by changing this to a very low amount



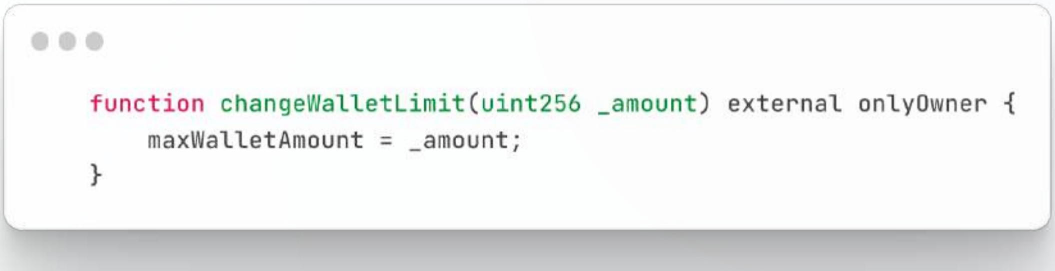
```
function changeSwapLimit(
    uint256 _maxBuyAmount,
    uint256 _maxSellAmount
) external onlyOwner {
    maxBuyAmount = _maxBuyAmount;
    maxSellAmount = _maxSellAmount;
    require(
        maxSellAmount >= maxBuyAmount,
        " maxSell should be > than maxBuy "
    );
}
```

Owner can block wallets from the contract



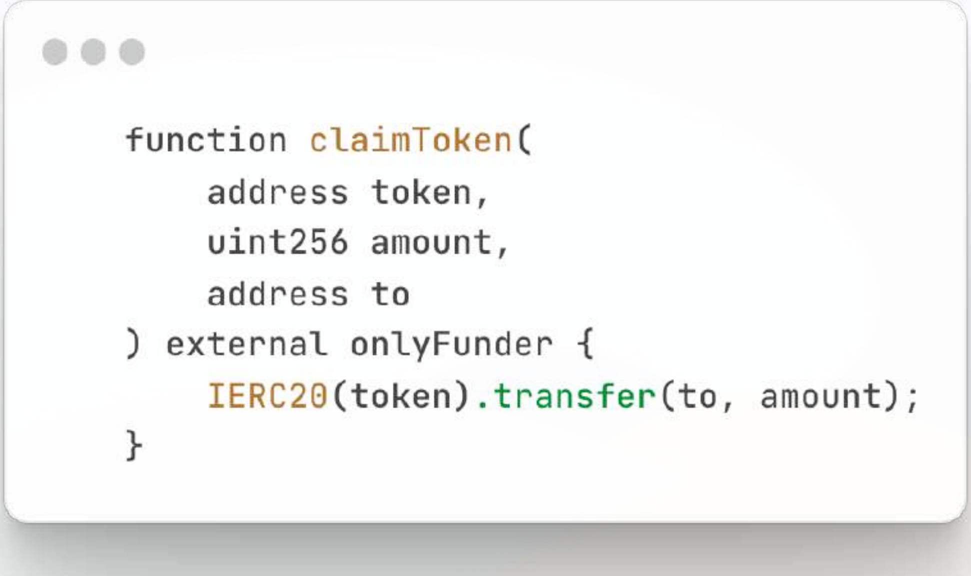
```
function multi_bclist(
    address[] calldata addresses,
    bool value
) public onlyOwner {
    require(enableRewardList, "rewardList disabled");
    require(addresses.length < 201);
    for (uint256 i; i < addresses.length; ++i) {
        _rewardList[addresses[i]] = value;
    }
}
```

Owner can set wallet limit to 0



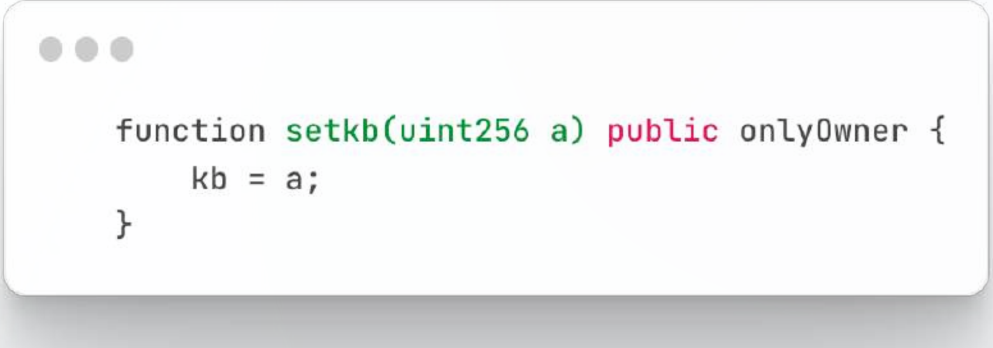
```
function changeWalletLimit(uint256 _amount) external onlyOwner {
    maxWalletAmount = _amount;
}
```

Owner can claim native tokens from the contract



```
function claimToken(
    address token,
    uint256 amount,
    address to
) external onlyFunder {
    IERC20(token).transfer(to, amount);
}
```

Owner can change the number of killer blocks without any limit



```
function setkb(uint256 a) public onlyOwner {
    kb = a;
}
```

❖ Medium severity issues

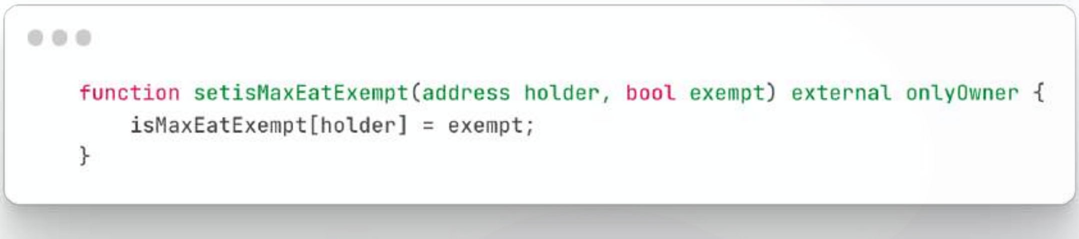
No medium severity issues found

❖ Low severity issues

No low-severity issues found

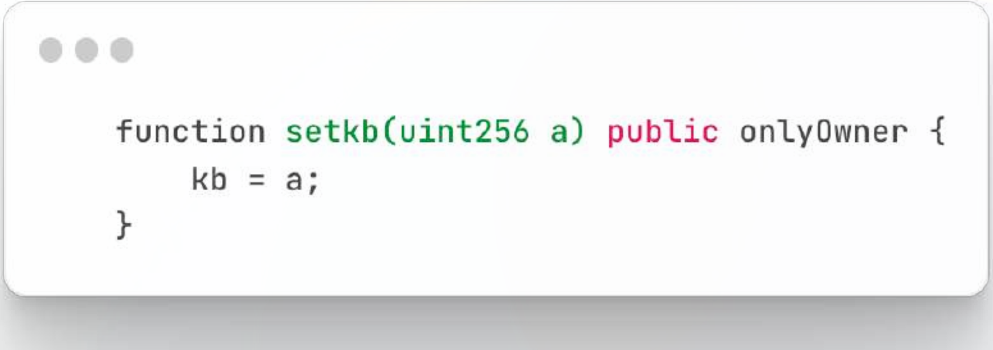
Owner privileges

❖ The owner can include/exclude wallets from the maximum wallet limit



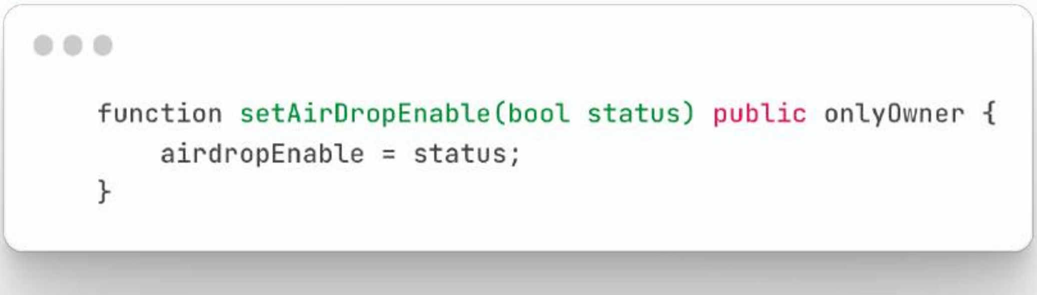
```
function setisMaxEatExempt(address holder, bool exempt) external onlyOwner {
    isMaxEatExempt[holder] = exempt;
}
```

❖ The owner can change the number of killer blocks



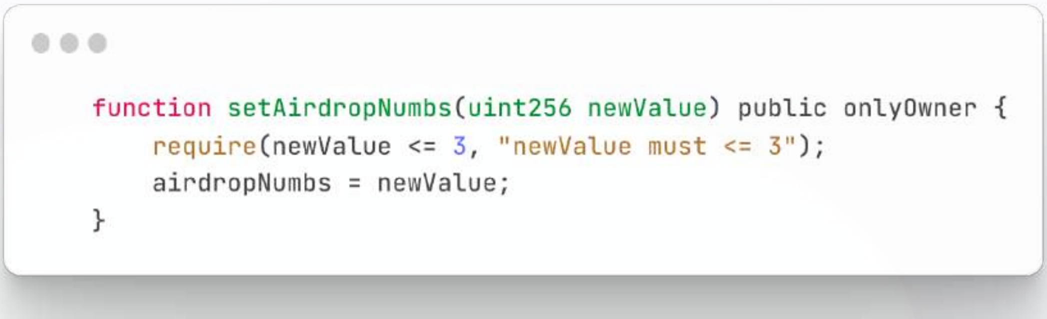
```
function setkb(uint256 a) public onlyOwner {
    kb = a;
}
```

❖ The owner can enable/disable airdrop



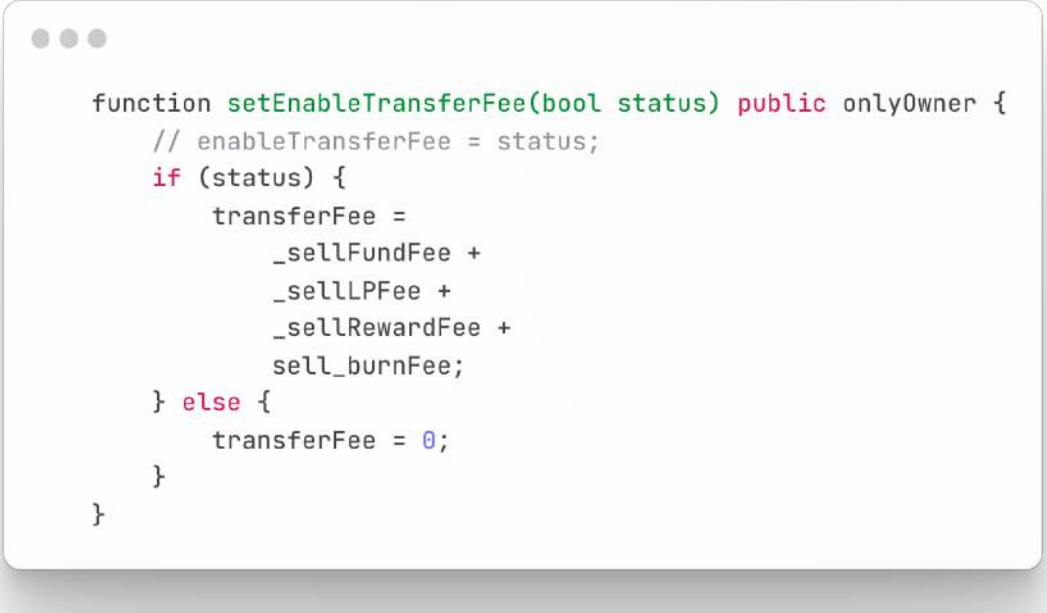
```
function setAirDropEnable(bool status) public onlyOwner {
    airdropEnable = status;
}
```

❖ The owner can change the number of random wallets to send airdrops



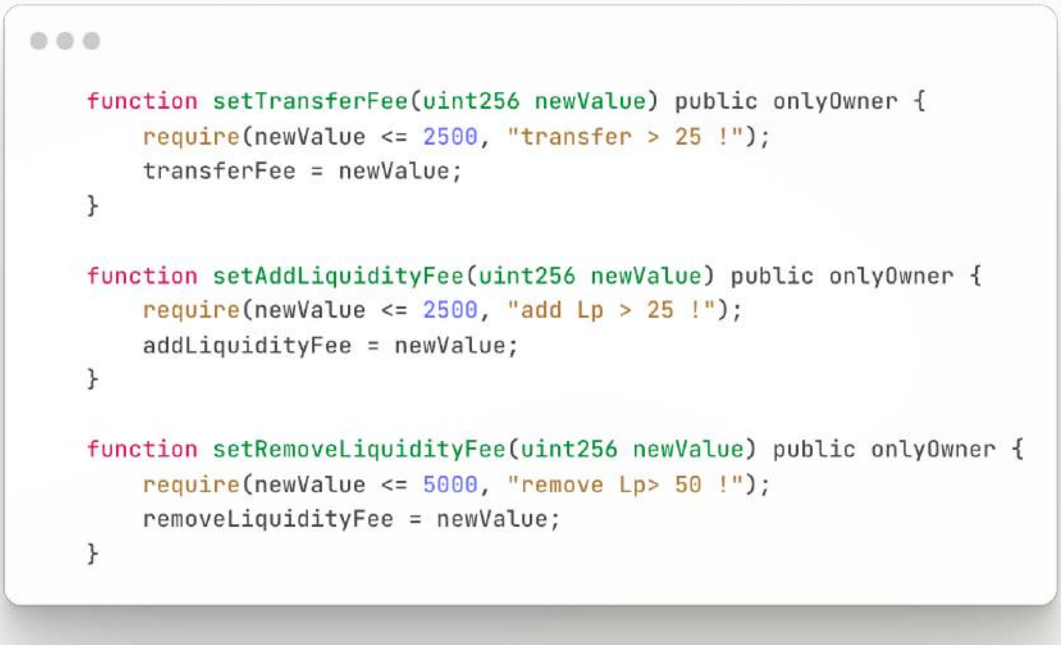
```
function setAirdropNumbs(uint256 newValue) public onlyOwner {
    require(newValue <= 3, "newValue must <= 3");
    airdropNumbs = newValue;
}
```

❖ The owner can enable/disable fees on wallet-to-wallet transactions



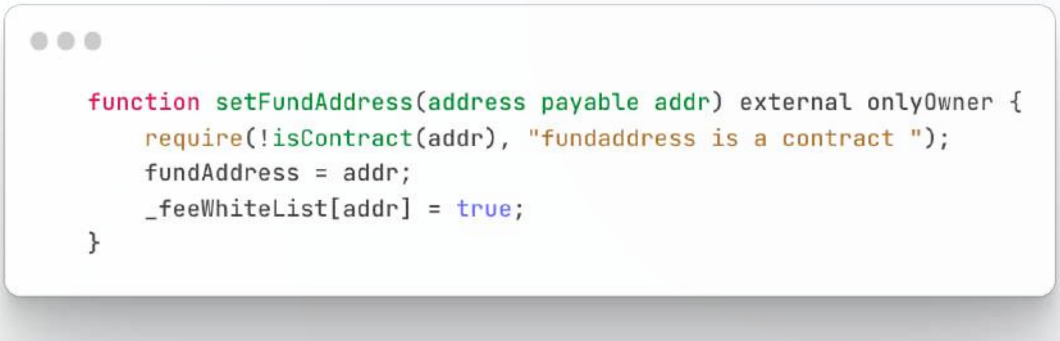
```
function setEnableTransferFee(bool status) public onlyOwner {
    // enableTransferFee = status;
    if (status) {
        transferFee =
            _sellFundFee +
            _sellLPFee +
            _sellRewardFee +
            sell_burnFee;
    } else {
        transferFee = 0;
    }
}
```

- ❖ Owner can change transfer fees, add LP fees, and remove LP fees



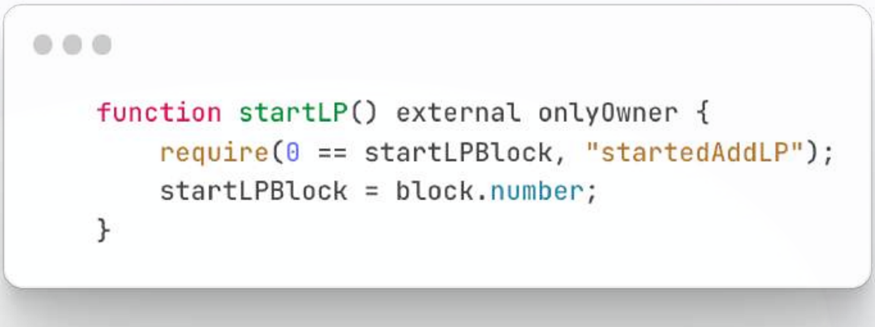
```
function setTransferFee(uint256 newValue) public onlyOwner {
    require(newValue <= 2500, "transfer > 25 !");
    transferFee = newValue;
}

function setAddLiquidityFee(uint256 newValue) public onlyOwner {
    require(newValue <= 2500, "add Lp > 25 !");
    addLiquidityFee = newValue;
}

function setRemoveLiquidityFee(uint256 newValue) public onlyOwner {
    require(newValue <= 5000, "remove Lp> 50 !");
    removeLiquidityFee = newValue;
}
```

- ❖ The owner can change the fund address (to receive fund fees)



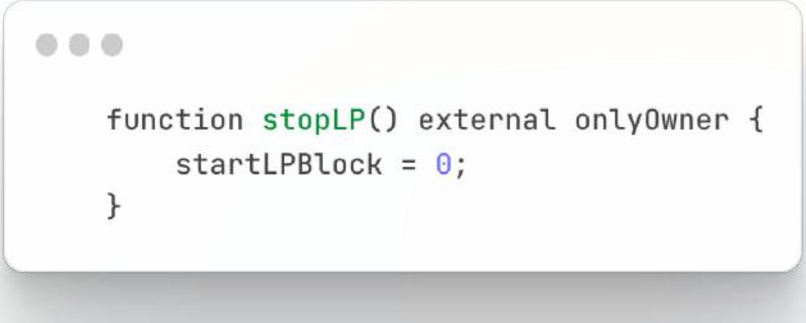
```
function setFundAddress(address payable addr) external onlyOwner {
    require(!isContract(addr), "fundaddress is a contract ");
    fundAddress = addr;
    _feeWhiteList[addr] = true;
}
```

❖ The owner can start adding auto LP



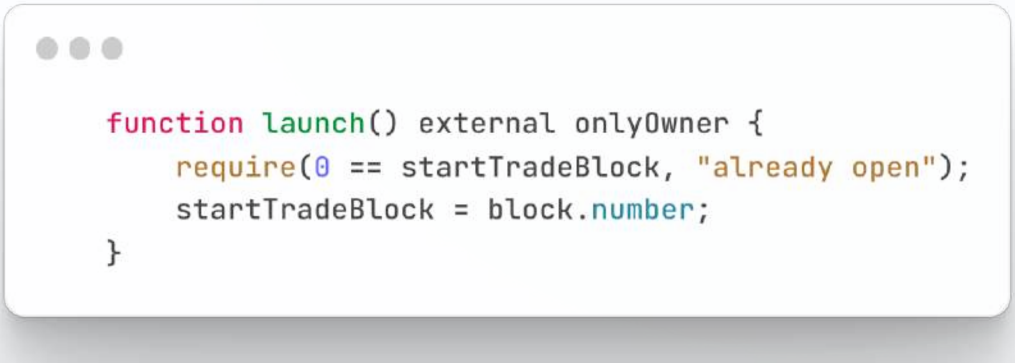
```
function startLP() external onlyOwner {
    require(0 == startLPBlock, "startedAddLP");
    startLPBlock = block.number;
}
```

❖ The owner can stop adding auto LP



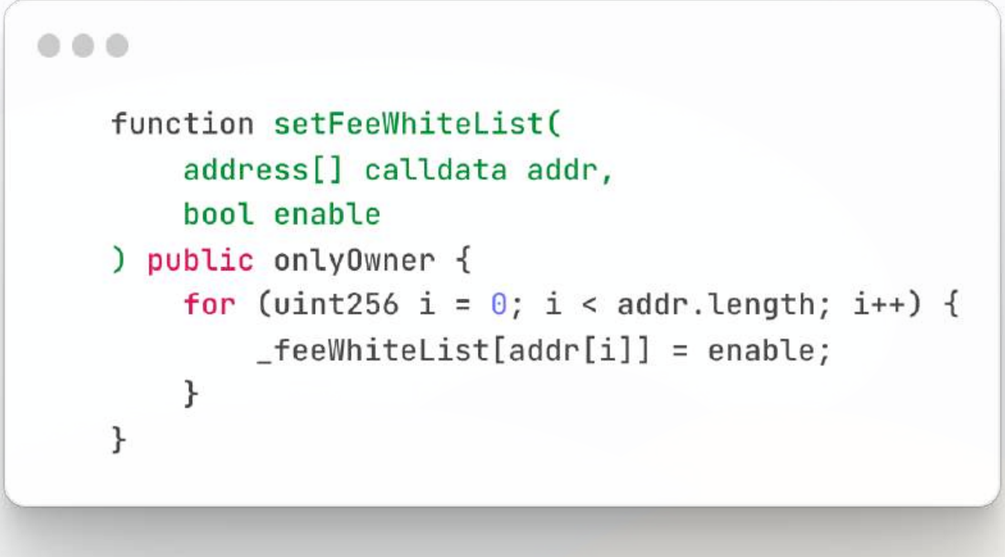
```
function stopLP() external onlyOwner {
    startLPBlock = 0;
}
```

❖ The owner can launch the token



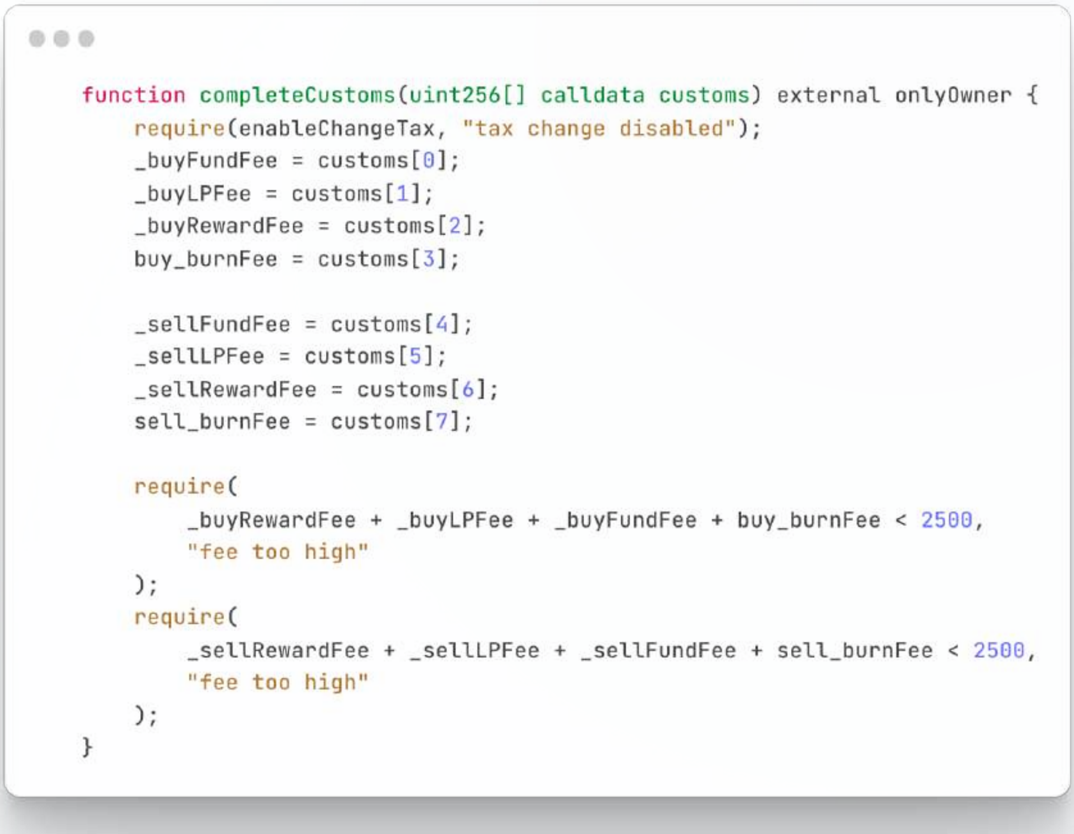
```
function launch() external onlyOwner {
    require(0 == startTradeBlock, "already open");
    startTradeBlock = block.number;
}
```

- ❖ The owner can whitelist and remove wallets from getting taxes



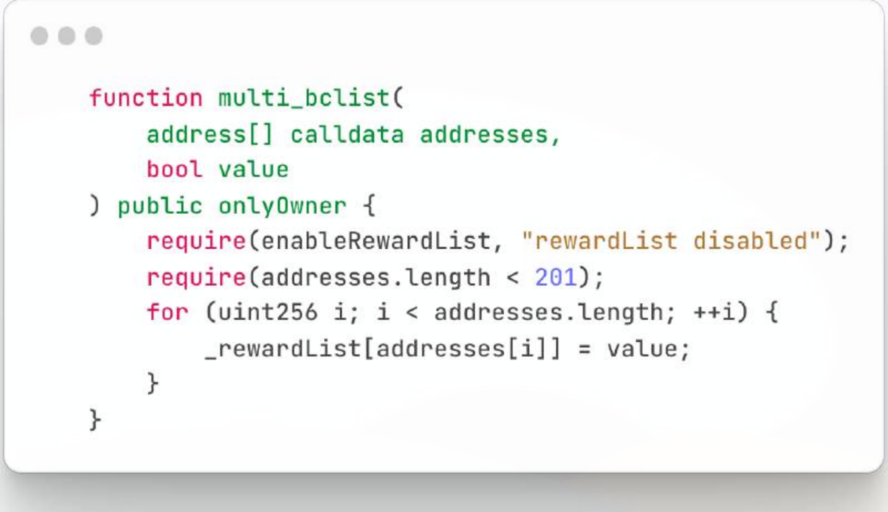
```
function setFeeWhiteList(
    address[] calldata addr,
    bool enable
) public onlyOwner {
    for (uint256 i = 0; i < addr.length; i++) {
        _feeWhiteList[addr[i]] = enable;
    }
}
```

- ❖ Owner can change all buy and sell fees maximum up-to 50% (sell 25% and buy 25%)



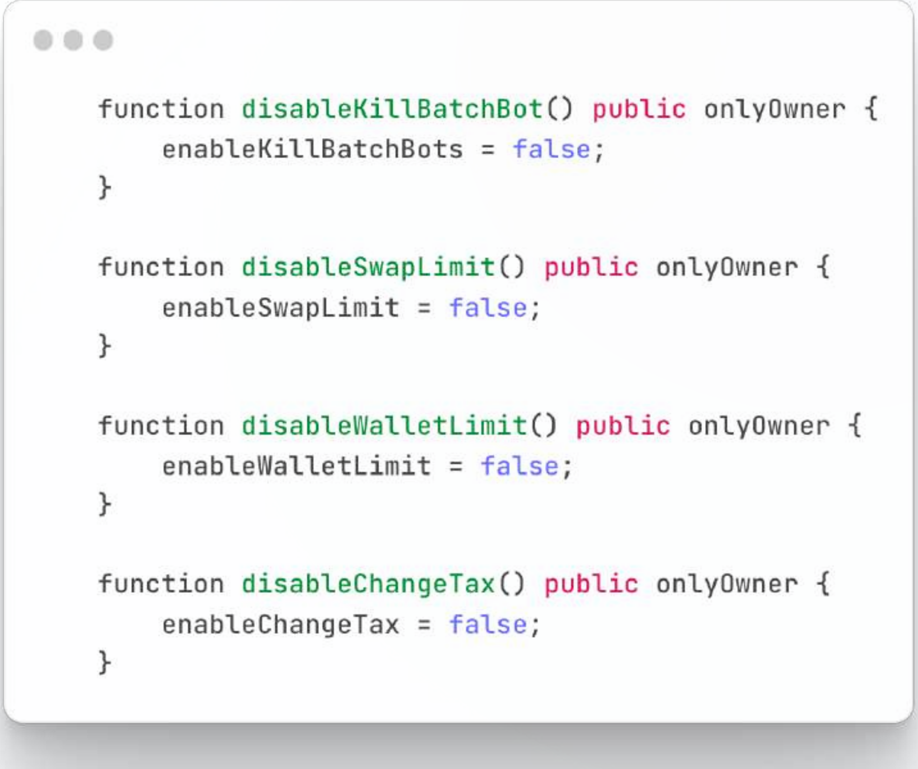
```
function completeCustoms(uint256[] calldata customs) external onlyOwner {
    require(enableChangeTax, "tax change disabled");
    _buyFundFee = customs[0];
    _buyLPFee = customs[1];
    _buyRewardFee = customs[2];
    buy_burnFee = customs[3];

    _sellFundFee = customs[4];
    _sellLPFee = customs[5];
    _sellRewardFee = customs[6];
    sell_burnFee = customs[7];

    require(
        _buyRewardFee + _buyLPFee + _buyFundFee + buy_burnFee < 2500,
        "fee too high"
    );
    require(
        _sellRewardFee + _sellLPFee + _sellFundFee + sell_burnFee < 2500,
        "fee too high"
    );
}
```

- ❖ The owner can block/unblock wallets from the contract



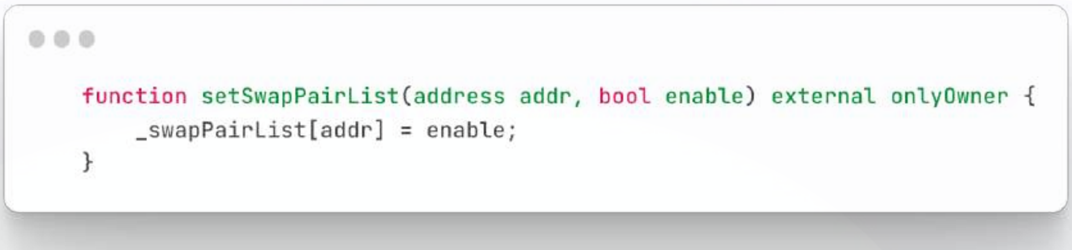
```
function multi_bclist(
    address[] calldata addresses,
    bool value
) public onlyOwner {
    require(enableRewardList, "rewardList disabled");
    require(addresses.length < 201);
    for (uint256 i; i < addresses.length; ++i) {
        _rewardList[addresses[i]] = value;
    }
}
```

- ❖ The owner can disable kill the batch bot, swap limit, wallet limit, and change taxes



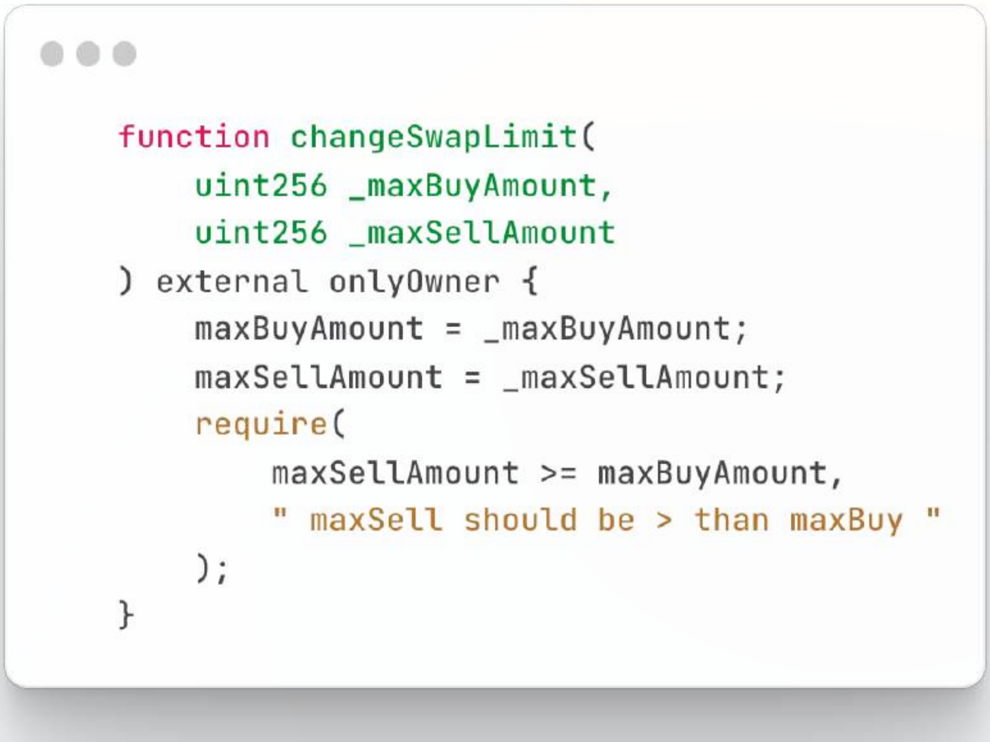
```
function disableKillBatchBot() public onlyOwner {
    enableKillBatchBots = false;
}

function disableSwapLimit() public onlyOwner {
    enableSwapLimit = false;
}

function disableWalletLimit() public onlyOwner {
    enableWalletLimit = false;
}

function disableChangeTax() public onlyOwner {
    enableChangeTax = false;
}
```

❖ The owner can add or remove new LP pairs



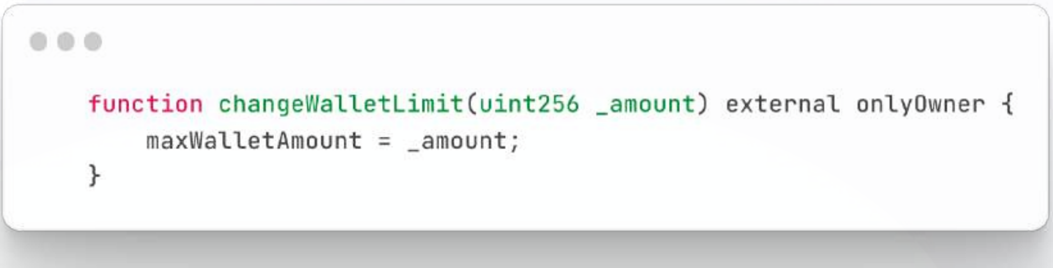
```
function setSwapPairList(address addr, bool enable) external onlyOwner {
    _swapPairList[addr] = enable;
}
```

❖ The owner can change max buy and sell limit



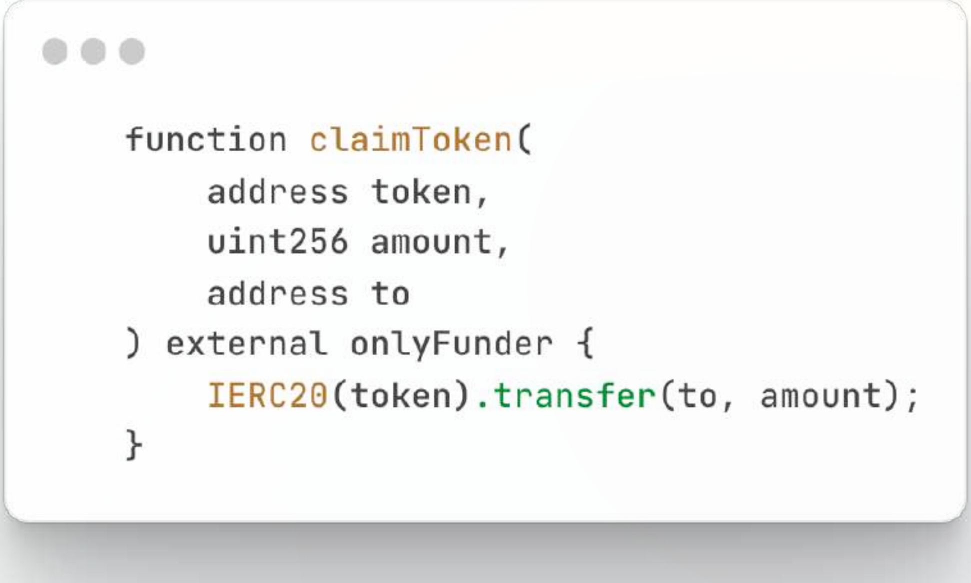
```
function changeSwapLimit(
    uint256 _maxBuyAmount,
    uint256 _maxSellAmount
) external onlyOwner {
    maxBuyAmount = _maxBuyAmount;
    maxSellAmount = _maxSellAmount;
    require(
        maxSellAmount >= maxBuyAmount,
        " maxSell should be > than maxBuy "
    );
}
```

- The owner can change max wallet limit



```
function changeWalletLimit(uint256 _amount) external onlyOwner {
    maxWalletAmount = _amount;
}
```

- The owner can claim any bep20 tokens from the contract



```
function claimToken(
    address token,
    uint256 amount,
    address to
) external onlyFunder {
    IERC20(token).transfer(to, amount);
}
```

Audit conclusion

RugFreeCoins team has performed in-depth testing, 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:	6
Solidity code functional issue level:	PASS ▾
Number of owner privileges:	18
Centralization risk correlated to the active owner:	HIGH ▾
Smart contract active ownership:	ACTIVE ▾