RUGFREECOINS

Elemental Betting Token

RugfreeCoins Verified on November 2nd, 2023

Overview

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

✅ The owner can’t 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't change fees over 10%..

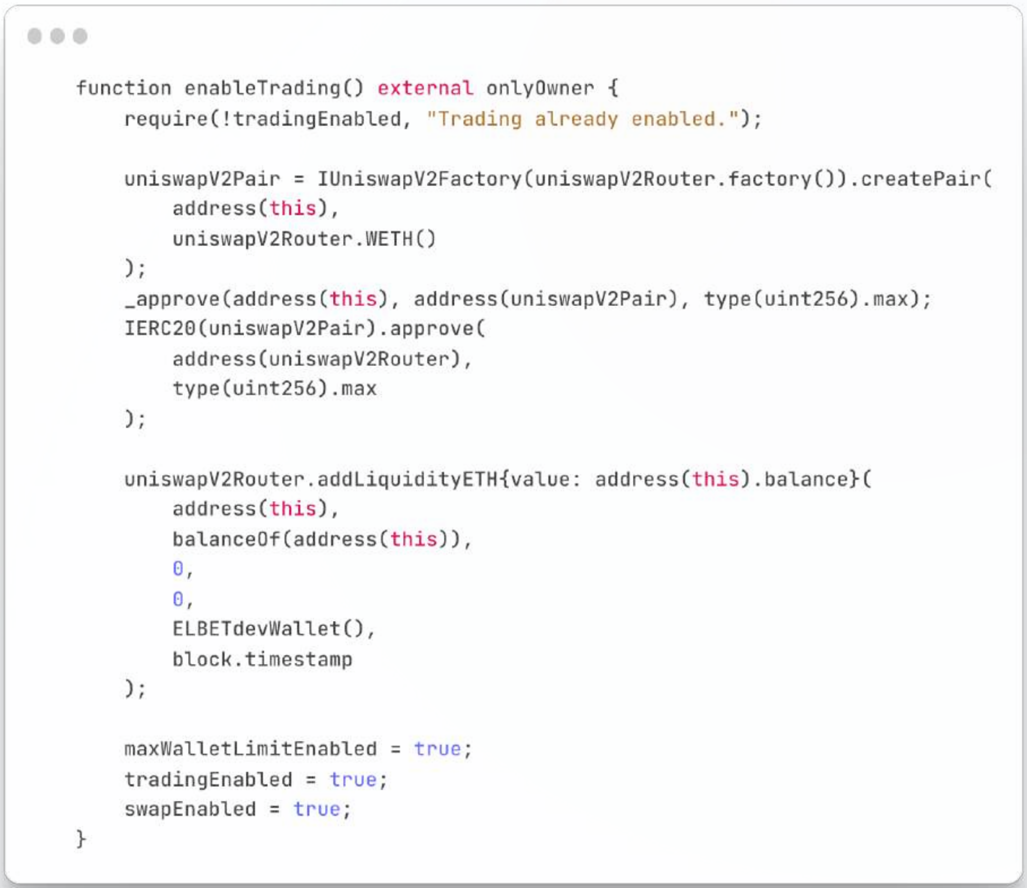
✅ The owner can't blacklist wallets.

✅ The owner can’t set a max wallet limit,it’s default set to 2%

❌ The owner can claim the contract’s balance of its own token.

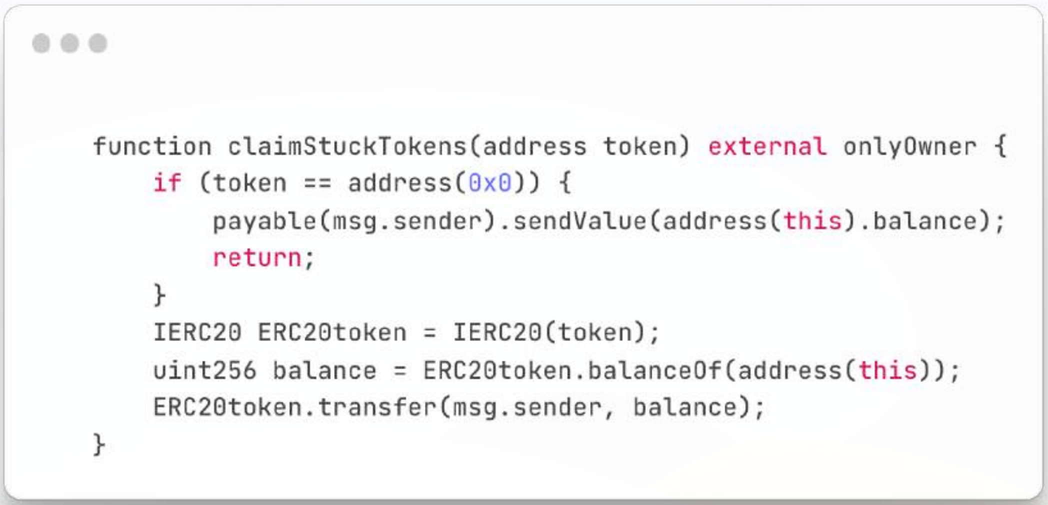
❗ HIGH SEVERITY ISSUES

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



```
function enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");

    uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(
        address(this),
        uniswapV2Router.WETH()
    );
    _approve(address(this), address(uniswapV2Pair), type(uint256).max);
    IERC20(uniswapV2Pair).approve(
        address(uniswapV2Router),
        type(uint256).max
    );

    uniswapV2Router.addLiquidityETH{value: address(this).balance}(
        address(this),
        balanceOf(address(this)),
        0,
        0,
        ELBETdevWallet(),
        block.timestamp
    );

    maxWalletLimitEnabled = true;
    tradingEnabled = true;
    swapEnabled = true;
}
```

The owner can claim native tokens from the contract



```
function claimStuckTokens(address token) external onlyOwner {
    if (token == address(0x0)) {
        payable(msg.sender).sendValue(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

Contents

Audit details

Audited project
Elemental Betting Token

Contract Address
0xf6FB594552246190435169614 5fa873167cb1C5C

Client contact
Elemental Betting Token Team

Blockchain
Ethereum

Project website
https://www.elbetcoin.com/

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – **please make sure to read it in full.**

DISCLAIMER

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. **This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.** No one shall have any right to rely on the report or its contents, and **RugfreeCoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (RugfreeCoins) owe no duty of care towards you or any other person**, nor does RugfreeCoins make any warranty or representation to any person on the accuracy or completeness of the report.

The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and RugfreeCoins hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report.

Except and only to the extent that it is prohibited by law, RugfreeCoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against RugfreeCoins, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

RugfreeCoins was commissioned by the **Elemental Betting Token Team** to perform an audit of the smart contract.

https://etherscan.io/address/0xf6FB594552246190435169614 5fa873167cb1C5C

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

▲ 5% tax when buying & selling

4% of trade goes to the marketing wallet in ETH
1% of trade goes to the Dev wallet in ETH

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 Elemental Betting token ecosystem.
- Anyone who’s interested in taking part in the future plans of Elemental Betting Token.
- Anyone who’s interested in making financial transactions with any other party using Elemental Betting 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	9 / 10
🛠️ Smart contract functionality assessment	9 / 10
🏆 **Total Score**	**8.2**/ 10

Contract details

Token contract details for 2nd of November 2023

Contract name	**Elemental Betting**
Contract address	**0xf6FB5945522461904351696145fa873167cb1C5C**
Token supply	**1,000,000,000**
Token ticker	**ELBET**
Decimals	**18**
Token holders	**2**
Transaction count	**2**
Contract deployer address	**0x1AEcBD387ab027430277184506c035a843514D3E**
Contract’s current owner address	**0x1AEcBD387ab027430277184506c035a843514D3E**
Marketing wallet	**0xAe393D57dfab1b0BCAa879101652803D9E49A616**
Dev wallet	**0x1AEcBD387ab027430277184506c035a843514D3E**

Contract code function details

№	Category	Item	Result
1	Coding conventions	ERC20 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	LOW
		Business implementation	HIGH
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
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 ❗

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 ❗
IUniswapV2 Factory	**Interface**			
L	feeTo	External ❗		NO ❗
L	feeToSetter	External ❗		NO ❗
L	getPair	External ❗		NO ❗
L	allPairs	External ❗		NO ❗
L	allPairsLength	External ❗		NO ❗
L	createPair	External ❗	🛑	NO ❗
L	setFeeTo	External ❗	🛑	NO ❗
L	setFeeToSetter	External ❗	🛑	NO ❗
Address	**Library**			
L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	🛑	
L	functionCall	Internal 🔒	🛑	
L	functionCall	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	functionStaticCall	Internal 🔒		

└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	verifyCallResultFromTarget	Internal 🔒		
└	verifyCallResult	Internal 🔒		
└	_revert	Private 🔐		
IERC20	**Interface**			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IERC20 Metadata	**Interface**	**IERC20**		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
ERC20	**Implementation**	**Context, IERC20, IERC20 Metadata**		
└		Public ❗	🛑	NO ❗
└	name	Public ❗		NO ❗

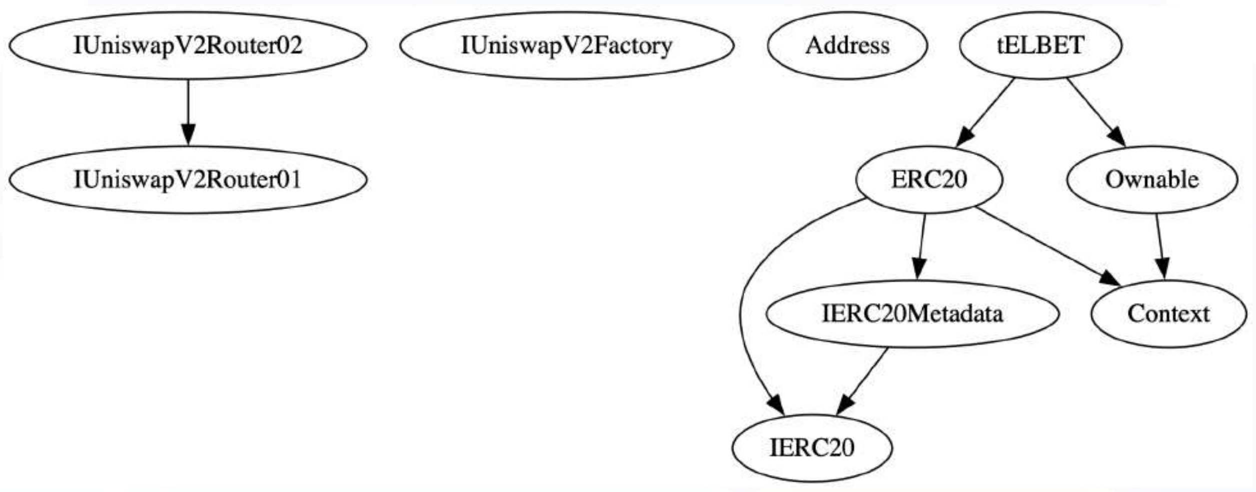
L	symbol	Public ❗		NO ❗
L	decimals	Public ❗		NO ❗
L	totalSupply	Public ❗		NO ❗
L	balanceOf	Public ❗		NO ❗
L	transfer	Public ❗	🛑	NO ❗
L	allowance	Public ❗		NO ❗
L	approve	Public ❗	🛑	NO ❗
L	transferFrom	Public ❗	🛑	NO ❗
L	increaseAllowance	Public ❗	🛑	NO ❗
L	decreaseAllowance	Public ❗	🛑	NO ❗
L	_transfer	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_approve	Internal 🔒	🛑	
L	_spendAllowance	Internal 🔒	🛑	
L	_beforeTokenTransfer	Internal 🔒	🛑	
L	_afterTokenTransfer	Internal 🔒	🛑	
Ownable	**Implementation**	**Context**		
L		Public ❗	🛑	NO ❗
L	owner	Public ❗		NO ❗
L	_checkOwner	Internal 🔒		
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	_transferOwnership	Internal 🔒	🛑	
tELBET	**Implementation**	**ERC20, Ownable**		
L		Public ❗	🛑	ERC20
L		External ❗	💵	NO ❗

└	burn	External ❗	🛑	NO ❗
└	RegisterPlayer	External ❗	🛑	NO ❗
└	SetinitialBetBalance	External ❗	🛑	onlyOwner
└	setEscrowContract	Public ❗	🛑	onlyOwner
└	claimStuckTokens	External ❗	🛑	onlyOwner
└	_checkELBETdev	Internal 🔒		
└	ELBETdevWallet	Public ❗		NO ❗
└	excludeFromFees	External ❗	🛑	onlyOwner
└	isExcludedFromFees	Public ❗		NO ❗
└	updateFees	External ❗	🛑	onlyOwner
└	changeELBETmktWallet	External ❗	🛑	onlyOwner
└	enableTrading	External ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	
└	setSwapEnabled	External ❗	🛑	onlyOwner
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	swapAndSendELBETmkt	Private 🔐	🛑	
└	setEnableMaxWalletLimit	External ❗	🛑	onlyOwner
└	excludeFromMaxWallet	External ❗	🛑	onlyOwner
└	isExcludedFromMaxWalletLimit	Public ❗		NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy

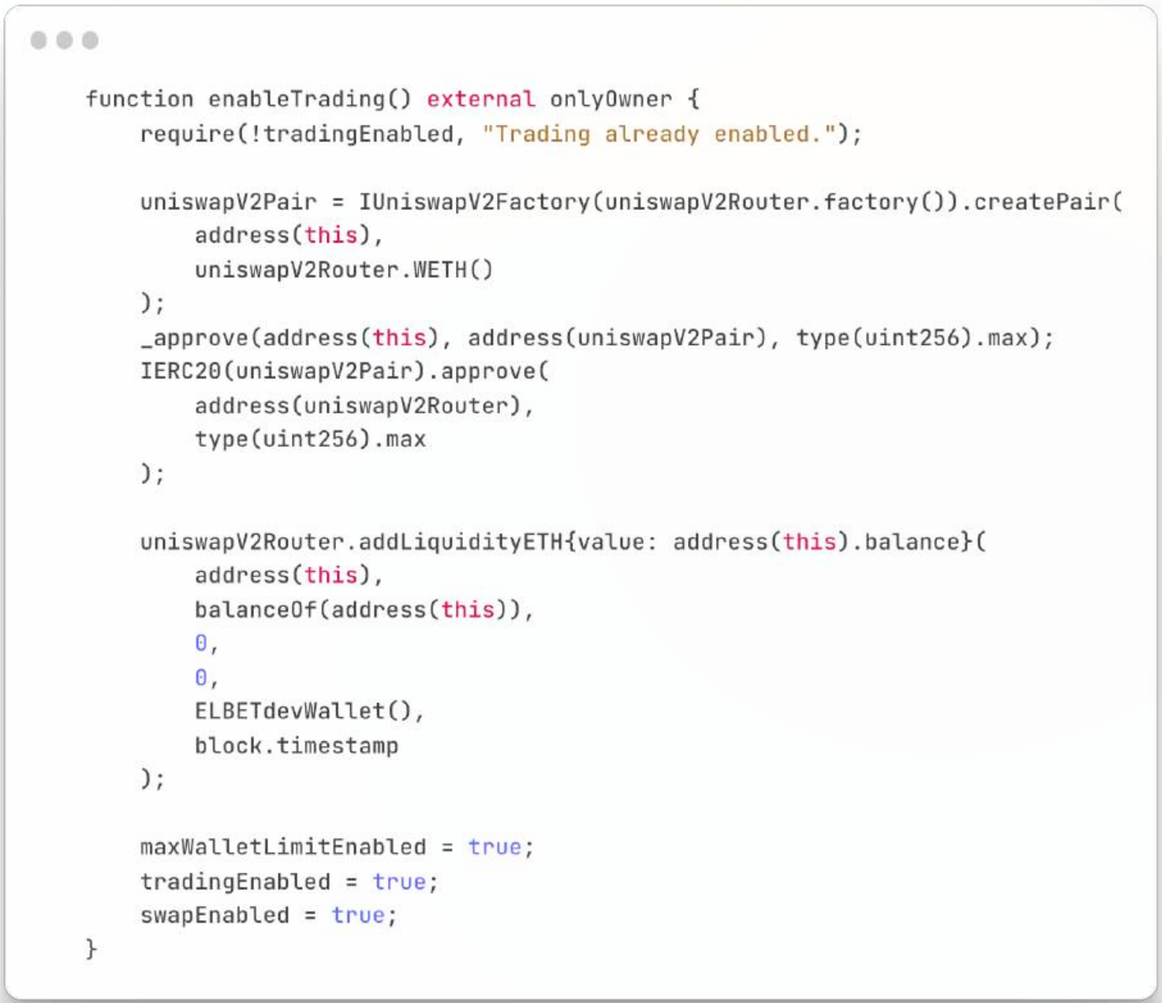


Security issue checking status

❖ High severity issues

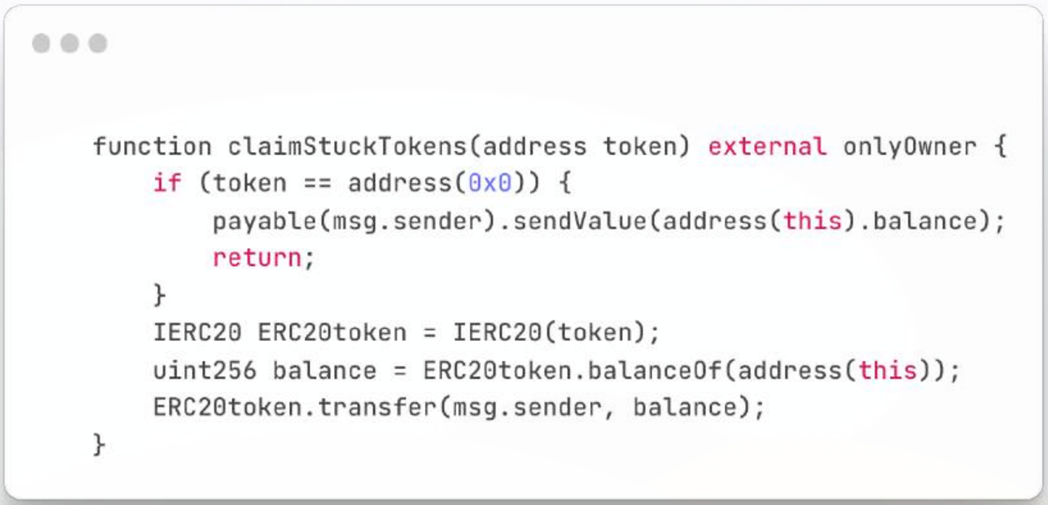
Inside the trade enable function, a token pair is created. If someone else has already created the pair before enabling trading, the trade enable function cannot be called again. This is because it attempts to create a pair that already exists, and as a result, this token cannot participate in a presale either due to this error.

It is advisable to create the pair in the constructor and remove it from the trade enable function to ensure perfect accuracy and avoid this issue.



```
function enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");

    uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(
        address(this),
        uniswapV2Router.WETH()
    );
    _approve(address(this), address(uniswapV2Pair), type(uint256).max);
    IERC20(uniswapV2Pair).approve(
        address(uniswapV2Router),
        type(uint256).max
    );

    uniswapV2Router.addLiquidityETH{value: address(this).balance}(
        address(this),
        balanceOf(address(this)),
        0,
        0,
        ELBETdevWallet(),
        block.timestamp
    );

    maxWalletLimitEnabled = true;
    tradingEnabled = true;
    swapEnabled = true;
}
```

The owner can claim native tokens from the contract



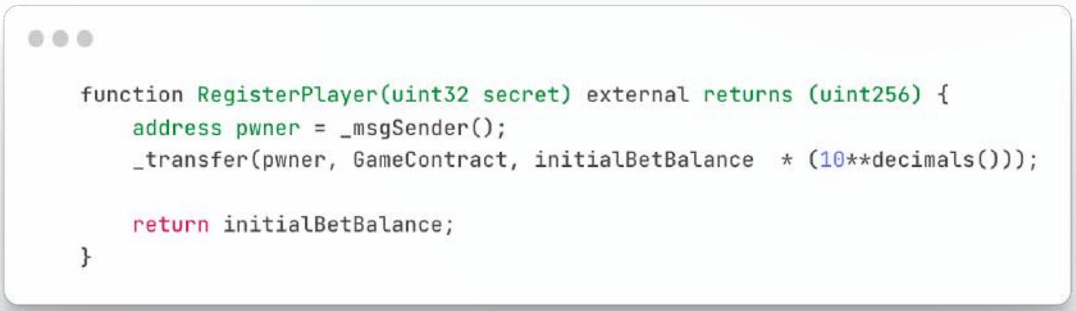
```
function claimStuckTokens(address token) external onlyOwner {
    if (token == address(0x0)) {
        payable(msg.sender).sendValue(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

❖ Medium severity issues

No medium severity issues found

❖ Low severity issues

In the "registerPlayer" function, it receives a "secret" parameter, but this parameter is never utilized within the function.



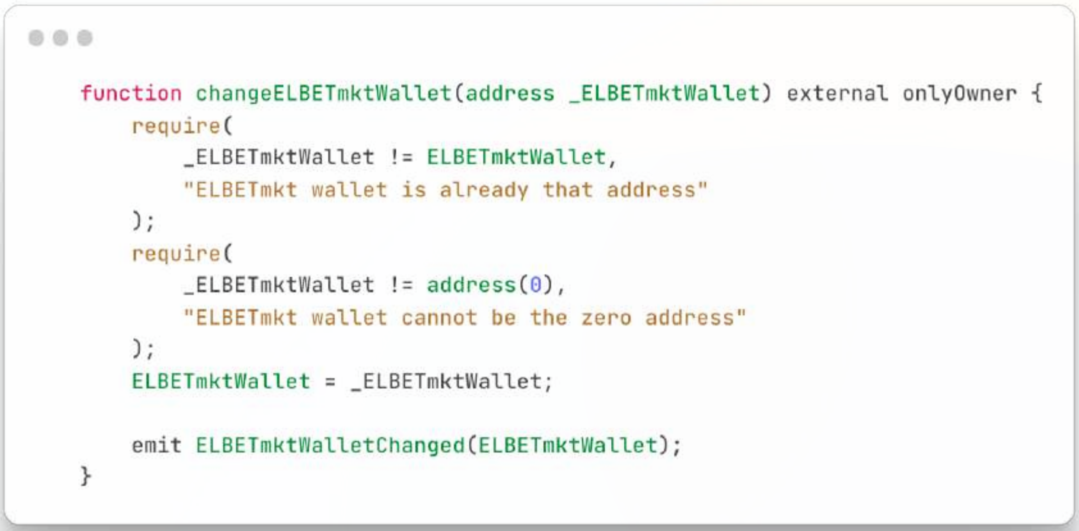
```
function RegisterPlayer(uint32 secret) external returns (uint256) {
    address pwner = _msgSender();
    _transfer(pwner, GameContract, initialBetBalance  * (10**decimals()));

    return initialBetBalance;
}
```

the contract declared its variables and mappings throughout the code

the contract declared its variables and mappings throughout the code, rather than in a single consolidated location. This approach can make the code harder to read and maintain. To improve the contract's readability and organization, it would be beneficial to consolidate all the variable declarations into a single location within the contract. By doing so, it becomes easier for developers to understand the structure of the contract and locate specific variables quickly. This organizational approach can also help prevent potential errors and improve the overall maintainability of the codebase

- **Informational**

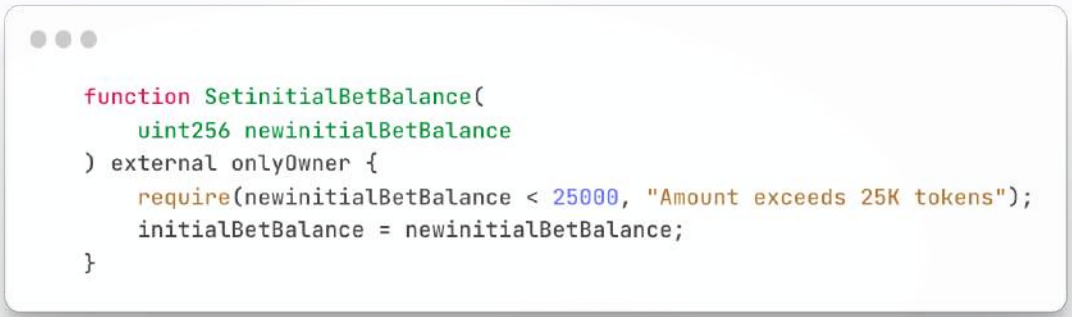
During the swap, the marketing wallet receives ETH. If the owner were to change the marketing wallet address to a multi-signature wallet or any other contract that cannot receive BNB, the swaps would fail. As a consequence of these swap failures, the selling process would also come to a halt.



```
function changeELBETmktWallet(address _ELBETmktWallet) external onlyOwner {
    require(
        _ELBETmktWallet != ELBETmktWallet,
        "ELBETmkt wallet is already that address"
    );
    require(
        _ELBETmktWallet != address(0),
        "ELBETmkt wallet cannot be the zero address"
    );
    ELBETmktWallet = _ELBETmktWallet;

    emit ELBETmktWalletChanged(ELBETmktWallet);
}
```

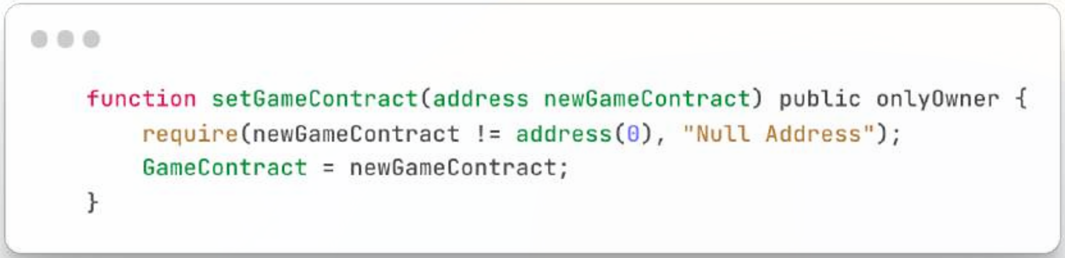
Owner privileges

❖ Owner can change initial bet value to maximum up-to 249999



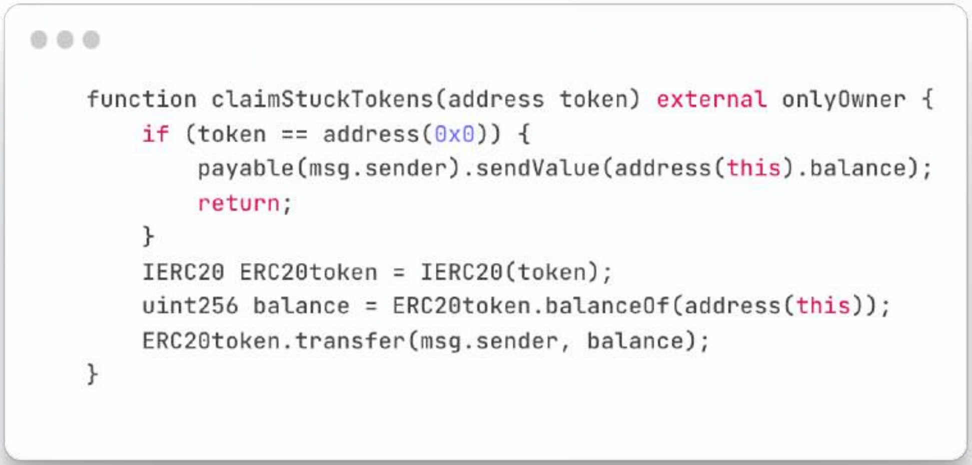
```
function SetinitialBetBalance(
    uint256 newinitialBetBalance
) external onlyOwner {
    require(newinitialBetBalance < 25000, "Amount exceeds 25K tokens");
    initialBetBalance = newinitialBetBalance;
}
```

❖ Owner can change Game contract address (which is not provided to auditing)



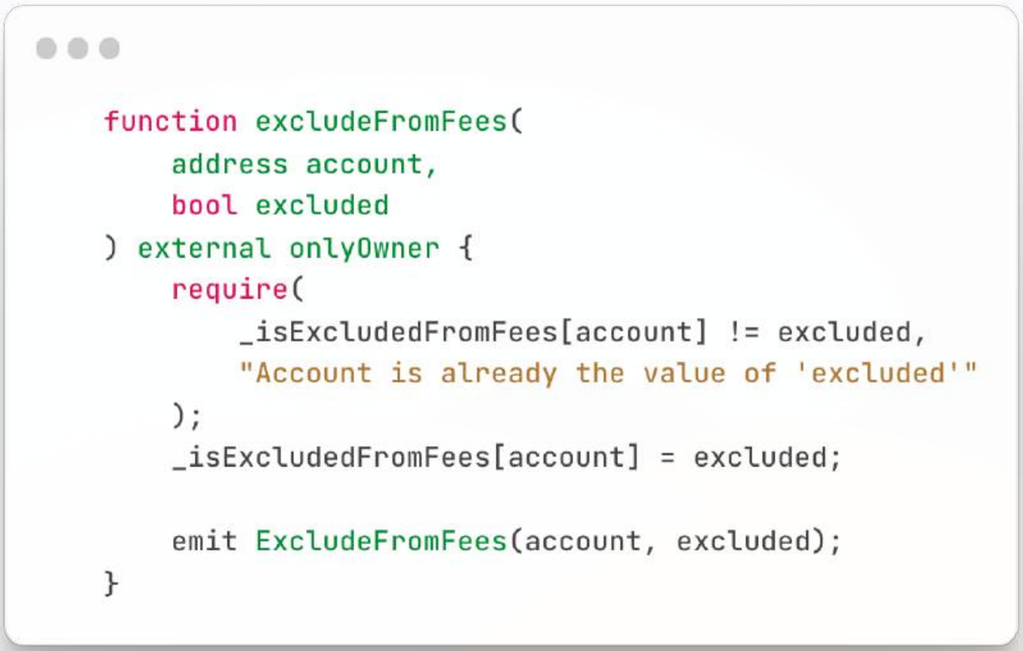
```
function setGameContract(address newGameContract) public onlyOwner {
    require(newGameContract != address(0), "Null Address");
    GameContract = newGameContract;
}
```

❖ Owner can claim ETH and and ERC20 token from the contract



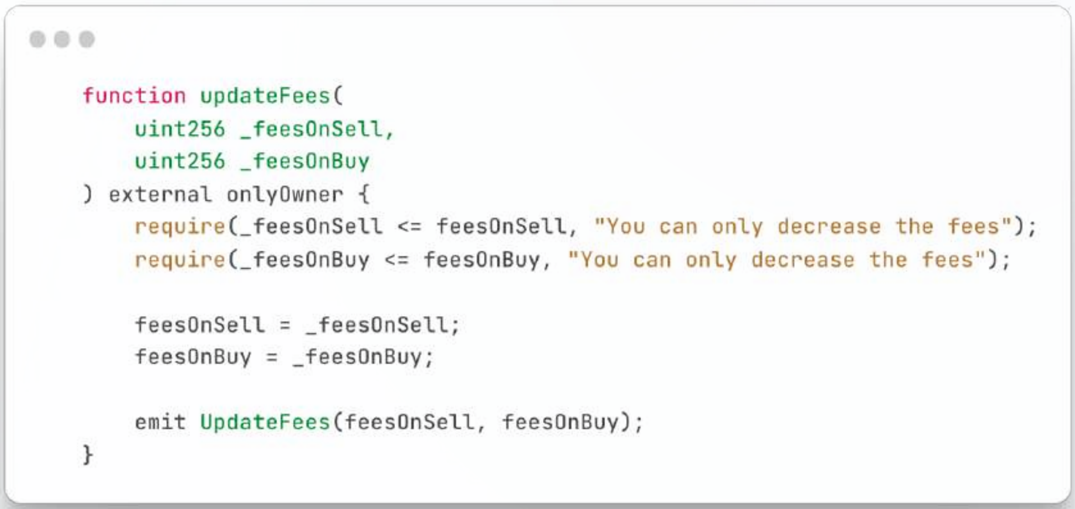
```
function claimStuckTokens(address token) external onlyOwner {
    if (token == address(0x0)) {
        payable(msg.sender).sendValue(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

❖ Owner can include/exclude wallets from the fees



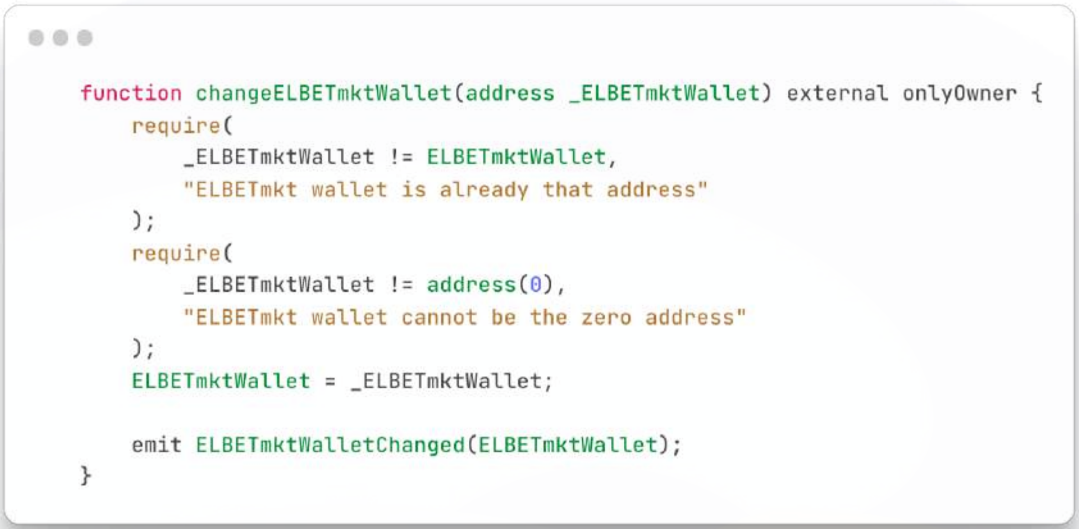
```
function excludeFromFees(
    address account,
    bool excluded
) external onlyOwner {
    require(
        _isExcludedFromFees[account] != excluded,
        "Account is already the value of 'excluded'"
    );
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}
```

❖ Owner can change buy and sell fees up-to 5%

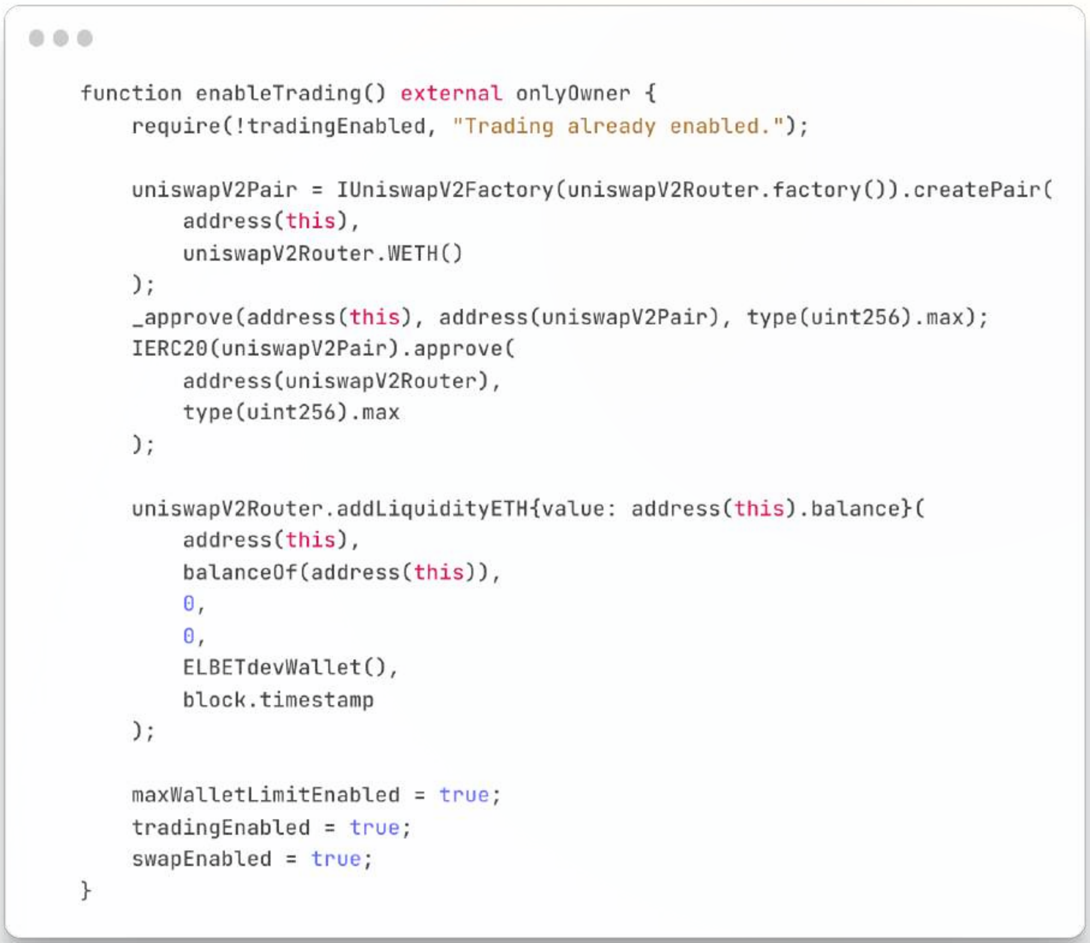


```
function updateFees(
    uint256 _feesOnSell,
    uint256 _feesOnBuy
) external onlyOwner {
    require(_feesOnSell <= feesOnSell, "You can only decrease the fees");
    require(_feesOnBuy <= feesOnBuy, "You can only decrease the fees");

    feesOnSell = _feesOnSell;
    feesOnBuy = _feesOnBuy;

    emit UpdateFees(feesOnSell, feesOnBuy);
}
```

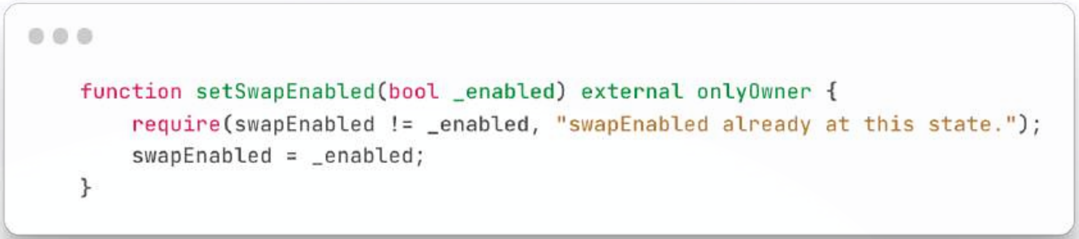
❖ Owner can change marketing wallet


```
function changeELBETmktWallet(address _ELBETmktWallet) external onlyOwner {
    require(
        _ELBETmktWallet != ELBETmktWallet,
        "ELBETmkt wallet is already that address"
    );
    require(
        _ELBETmktWallet != address(0),
        "ELBETmkt wallet cannot be the zero address"
    );
    ELBETmktWallet = _ELBETmktWallet;

    emit ELBETmktWalletChanged(ELBETmktWallet);
}
```

❖ Owner can enable trading,once enabled can not disable again

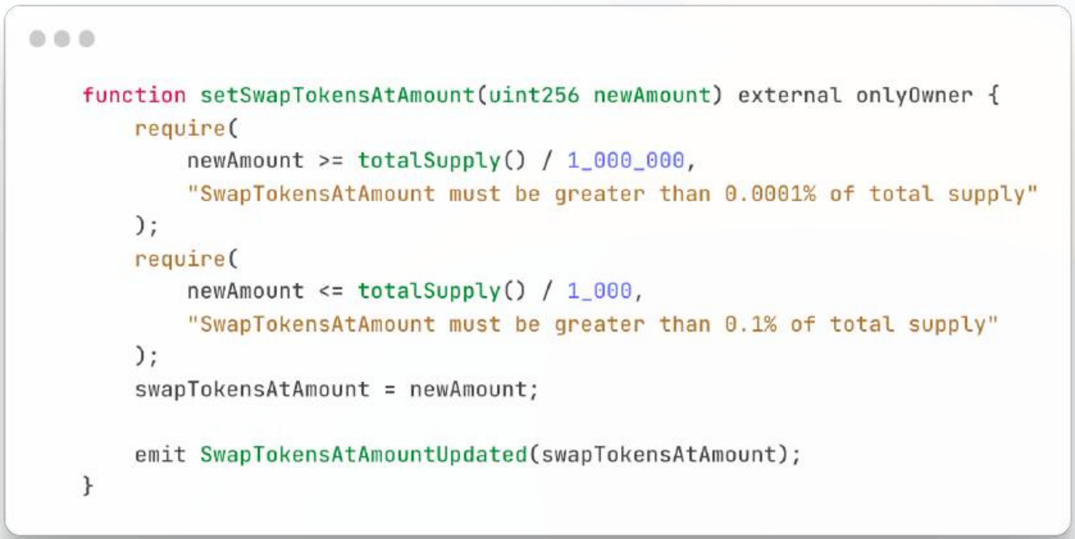

```
function enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");

    uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(
        address(this),
        uniswapV2Router.WETH()
    );
    _approve(address(this), address(uniswapV2Pair), type(uint256).max);
    IERC20(uniswapV2Pair).approve(
        address(uniswapV2Router),
        type(uint256).max
    );

    uniswapV2Router.addLiquidityETH{value: address(this).balance}(
        address(this),
        balanceOf(address(this)),
        0,
        0,
        ELBETdevWallet(),
        block.timestamp
    );

    maxWalletLimitEnabled = true;
    tradingEnabled = true;
    swapEnabled = true;
}
```

❖ Owner can enable/disable swap



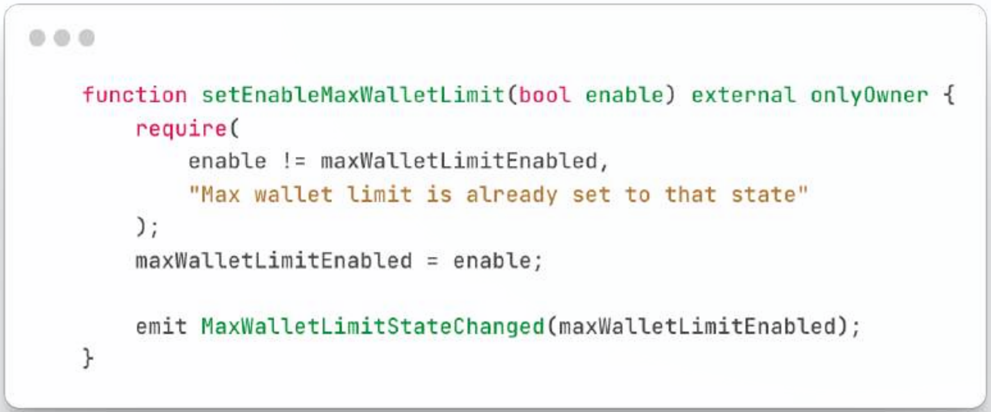
```
function setSwapEnabled(bool _enabled) external onlyOwner {
    require(swapEnabled != _enabled, "swapEnabled already at this state.");
    swapEnabled = _enabled;
}
```

❖ Owner can change swap point



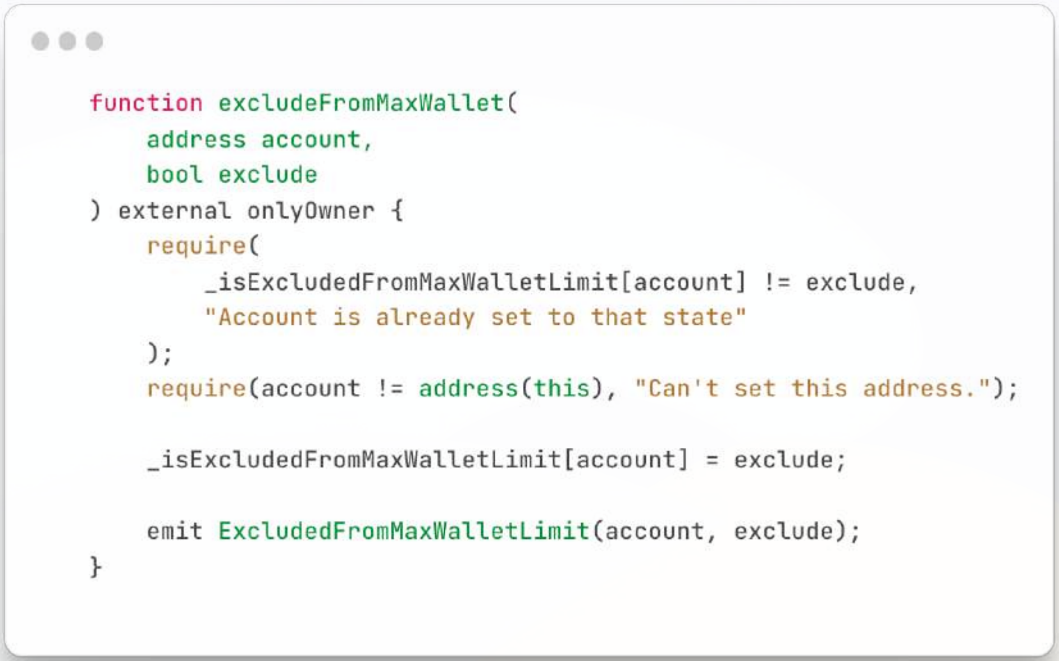
```
function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner {
    require(
        newAmount >= totalSupply() / 1_000_000,
        "SwapTokensAtAmount must be greater than 0.0001% of total supply"
    );
    require(
        newAmount <= totalSupply() / 1_000,
        "SwapTokensAtAmount must be greater than 0.1% of total supply"
    );
    swapTokensAtAmount = newAmount;

    emit SwapTokensAtAmountUpdated(swapTokensAtAmount);
}
```

❖ Owner can enable disable max wallet limit



```
function setEnableMaxWalletLimit(bool enable) external onlyOwner {
    require(
        enable != maxWalletLimitEnabled,
        "Max wallet limit is already set to that state"
    );
    maxWalletLimitEnabled = enable;

    emit MaxWalletLimitStateChanged(maxWalletLimitEnabled);
}
```

❖ Owner can include/exclude wallets from max wallet limit



```
function excludeFromMaxWallet(
    address account,
    bool exclude
) external onlyOwner {
    require(
        _isExcludedFromMaxWalletLimit[account] != exclude,
        "Account is already set to that state"
    );
    require(account != address(this), "Can't set this address.");

    _isExcludedFromMaxWalletLimit[account] = exclude;

    emit ExcludedFromMaxWalletLimit(account, exclude);
}
```

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:	4
Solidity code functional issue level:	PASS
Number of owner privileges:	11
Centralization risk correlated to the active owner:	HIGH
Smart contract active ownership:	ACTIVE