RugFreeCoins Audit

Babypepe 2.0 Token
Smart Contract Security Audit

July 9th ,2023

Overview

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

✅ The owner can’t pause trading

✅ The owner can't blacklist wallets.

✅ The owner can't set a max wallet limit

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

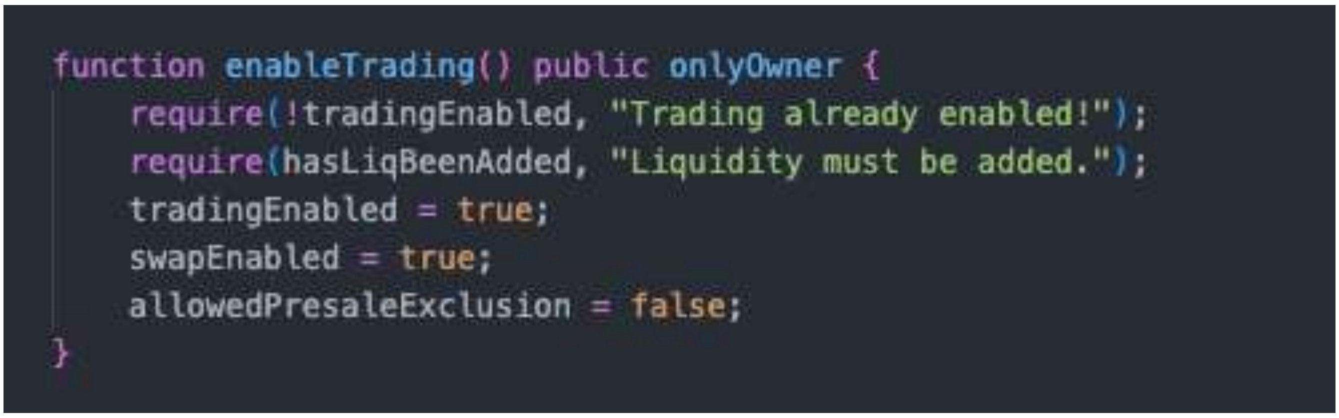
✅ The owner can't change fees by more than 20%.

✅ The owner can't set a max transaction limit

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

- **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() public onlyOwner {
    require(!tradingEnabled, "Trading already enabled!");
    require(hasLiqBeenAdded, "Liquidity must be added.");
    tradingEnabled = true;
    swapEnabled = true;
    allowedPresaleExclusion = false;
}
```

Contents

Audit details

Audited project
Babypepe 2.0 Token

Contract Address
0x6fA33cDf66cBb1ADbeB06C4d60cF3e6d45F13166

Client contact
Babypepe 2.0 Token Team

Blockchain
Binance Smart chain

Project website
http://babypepe20.baby

Disclaimer

Background

Rugfreecoins was commissioned by the Babypepe 2.0 Token Team to perform an audit of the smart contract.

https://bscscan.com/token/0x6fa33cdf66cbb1adbeb06c4d60cf3e6d45f13166#code

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

10% tax when buying & selling

- 10% of trade goes to the dev wallet in BNB

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 Babypepe 2.0 token ecosystem.
- Anyone who’s interested in taking part in the future plans of Babypepe 2.0 Token.
- Anyone who’s interested in making financial transactions with any other party using Babypepe 2.0 Token as the currency.

Potential to grow with score points

1.	Project efficiency	8/10
2.	Project uniqueness	8/10
3	Information quality	8/10
4	Service quality	8/10
5	System quality	8/10
6	Impact on the community	8/10
7	Impact on the business	9/10
8	Preparing for the future	8/10
9	Smart contract security	9/10
10	Smart contract functionality assessment	9/10
Total Points		**8.3/10**

Contract details

Token contract details for 9th of July 2023

Contract name	Babypepe 2.0
Contract address	0x6fA33cDf66cBb1ADbeB06C4d60cF3e6d45F13166
Token supply	420,000,000,000,000,000
Token ticker	Babypepe2.0
Decimals	6
Token holders	1
Transaction count	1
Contract deployer address	0xA889C06Bb24c382790955DC20CC423F2b03672c0
Contract's current owner address	0xA889C06Bb24c382790955DC20CC423F2b03672c0

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	Low issue
		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 & centralization	Access control of owners	High Issue
		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
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
IERC20	**Interface**			
└	totalSupply	External ❗		NO❗
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	getOwner	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗

└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IFactoryV2	**Interface**			
└	getPair	External ❗		NO❗
└	createPair	External ❗	🛑	NO❗
IV2Pair	**Interface**			
└	factory	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	sync	External ❗	🛑	NO❗
IRouter01	**Interface**			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	swapExactETHForTokens	External ❗	💵	NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗

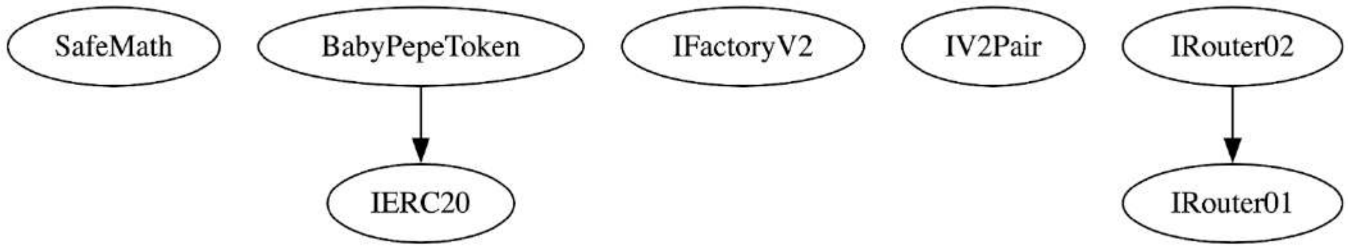
IRouter02	**Interface**	**IRouter01**		
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokens	External ❗	🛑	NO ❗
BabyPepe Token	**Implementation**	**IERC20**		
└		Public ❗	💵	NO ❗
└		External ❗	💵	NO ❗
└	totalSupply	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	name	External ❗		NO ❗
└	getOwner	External ❗		NO ❗
└	allowance	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	approve	External ❗	🛑	NO ❗
└	_approve	Internal 🔒	🛑	
└	approveContractContingency	External ❗	🛑	onlyOwner
└	transferFrom	External ❗	🛑	NO ❗

└	_hasLimits	Internal 🔒		
└	_transfer	Internal 🔒	🛑	
└	finalizeTransfer	Internal 🔒	🛑	
└	_transferAmount	Internal 🔒	🛑	
└	_checkLiquidityAdd	Internal 🔒	🛑	
└	swapTokensForEth	Private 🔐	🛑	lockThe Swap
└	sendETHToFee	Private 🔐	🛑	
└	manualswap	External ❗	🛑	NO❗
└	manualsend	External ❗	🛑	NO❗
└	transferOwner	External ❗	🛑	onlyOwner
└	renounceOwnership	External ❗	🛑	onlyOwner
└	excludePresaleAddresses	External ❗	🛑	onlyOwner
└	setDevAddress	External ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	enableTrading	Public ❗	🛑	onlyOwner
└	setFee	Public ❗	🛑	onlyOwner
└	updateSwapEnabled	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

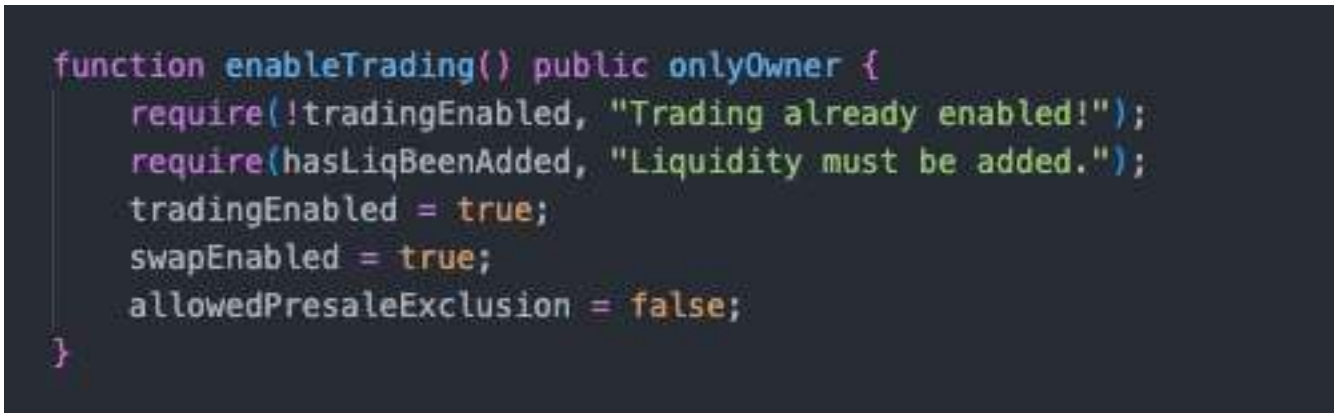
Inheritance Hierarchy



Security issue checking status

❖ **High severity issues**

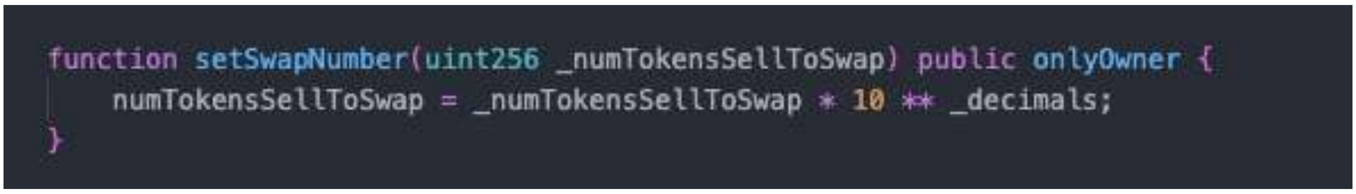
The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.



```
function enableTrading() public onlyOwner {
    require(!tradingEnabled, "Trading already enabled!");
    require(hasLiqBeenAdded, "Liquidity must be added.");
    tradingEnabled = true;
    swapEnabled = true;
    allowedPresaleExclusion = false;
}
```

The owner can change the swap token amount without minimum value, the owner can stop trading by setting this to 0

Informed & Fixed

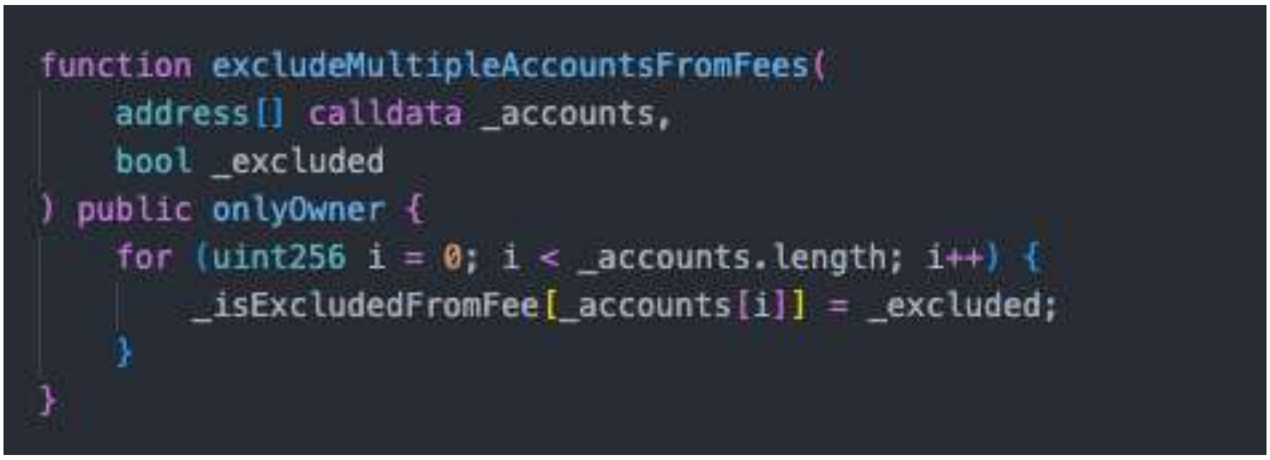


```
function setSwapNumber(uint256 _numTokensSellToSwap) public onlyOwner {
    numTokensSellToSwap = _numTokensSellToSwap * 10 ** _decimals;
}
```

❖ **Medium severity issues**
No medium severity issues found

❖ Low severity issues

When the owner excludes multiple accounts from fees there is no limitation, if the owner excludes large numbers of wallets at a time this function can fail due to out of gas



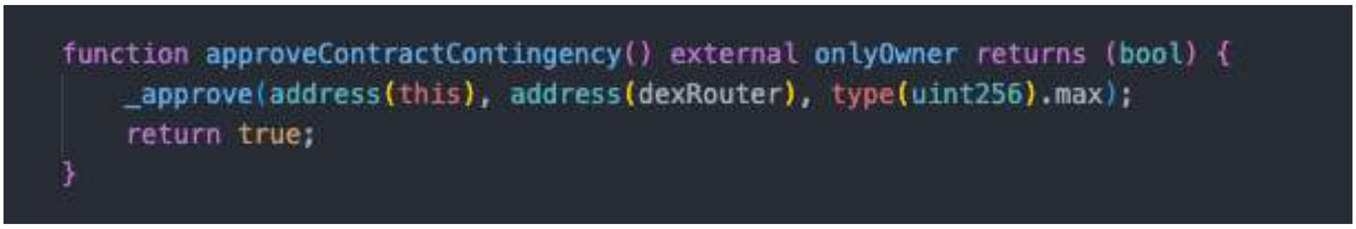
```
function excludeMultipleAccountsFromFees(
    address[] calldata _accounts,
    bool _excluded
) public onlyOwner {
    for (uint256 i = 0; i < _accounts.length; i++) {
        _isExcludedFromFee[_accounts[i]] = _excluded;
    }
}
```

❖ Centralization Risk

No Centralization issues found

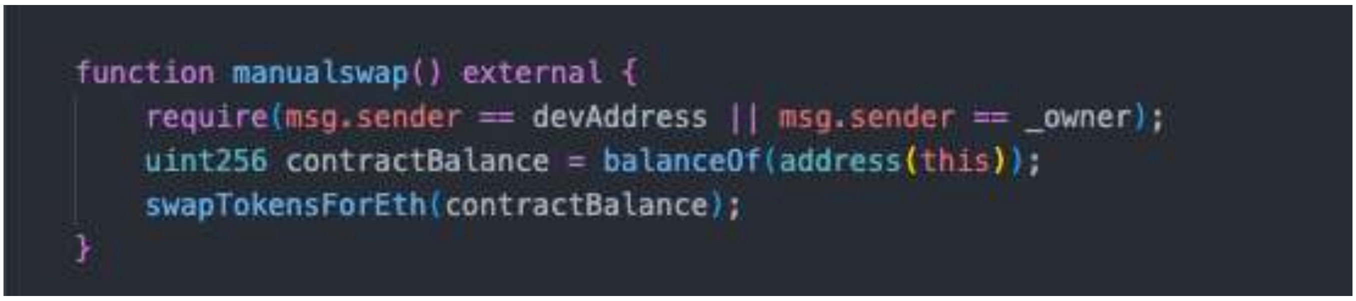
Owner privileges

❖ The owner can give approval to the router address to spend the max amount of tokens from the contract



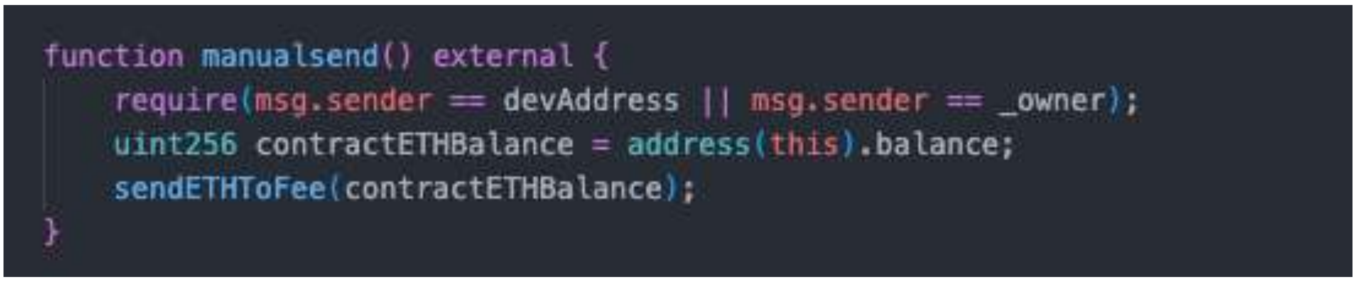
```
function approveContractContingency() external onlyOwner returns (bool) {
    _approve(address(this), address(dexRouter), type(uint256).max);
    return true;
}
```

❖ The owner can dev wallet can manually trigger the swapping



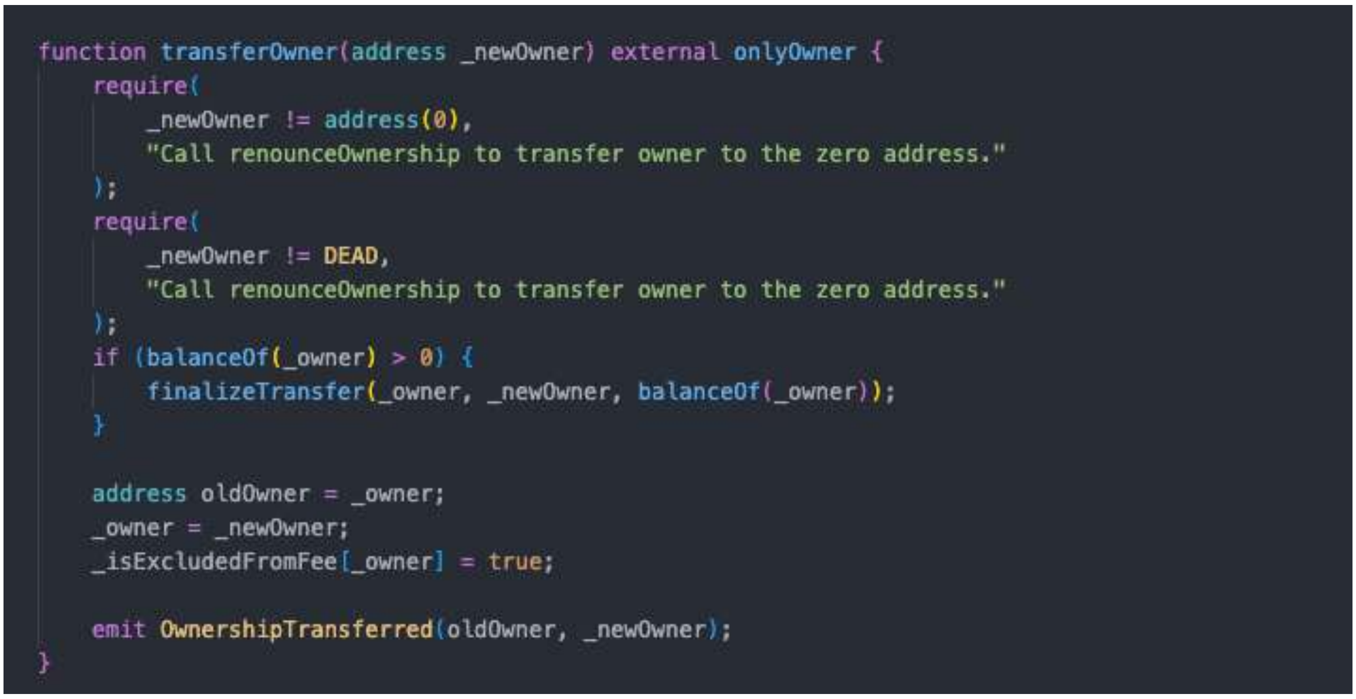
```
function manualswap() external {
    require(msg.sender == devAddress || msg.sender == _owner);
    uint256 contractBalance = balanceOf(address(this));
    swapTokensForEth(contractBalance);
}
```

❖ The owner and dev wallet can send contract BNB to dev address



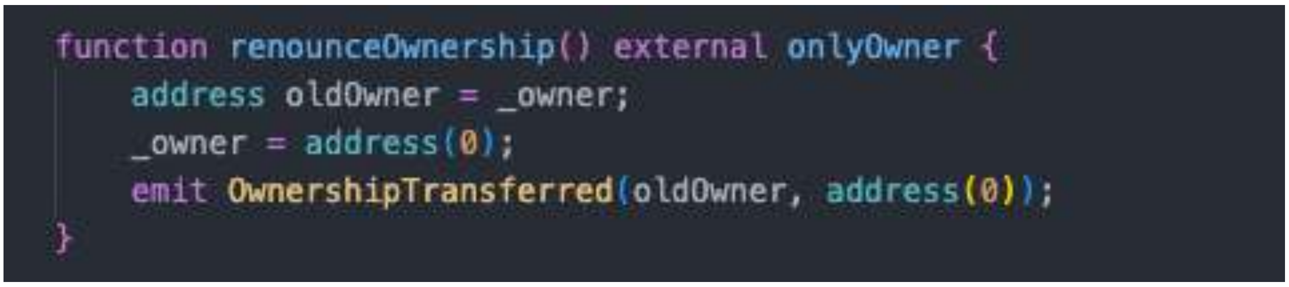
```
function manualsend() external {
    require(msg.sender == devAddress || msg.sender == _owner);
    uint256 contractETHBalance = address(this).balance;
    sendETHToFee(contractETHBalance);
}
```

❖ The owner can transfer the ownership



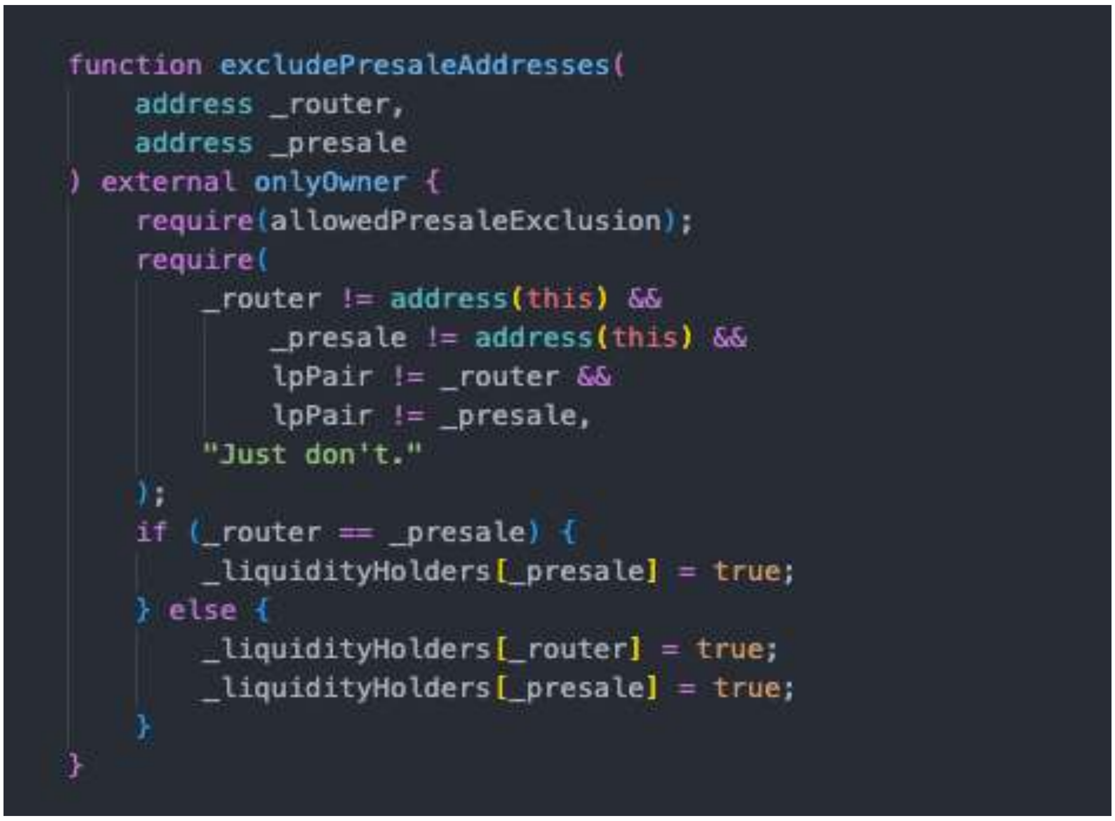
```
function transferOwner(address _newOwner) external onlyOwner {
    require(
        _newOwner != address(0),
        "Call renounceOwnership to transfer owner to the zero address."
    );
    require(
        _newOwner != DEAD,
        "Call renounceOwnership to transfer owner to the zero address."
    );
    if (balanceOf(_owner) > 0) {
        finalizeTransfer(_owner, _newOwner, balanceOf(_owner));
    }

    address oldOwner = _owner;
    _owner = _newOwner;
    _isExcludedFromFee[_owner] = true;

    emit OwnershipTransferred(oldOwner, _newOwner);
}
```

❖ The owner can renounce ownership



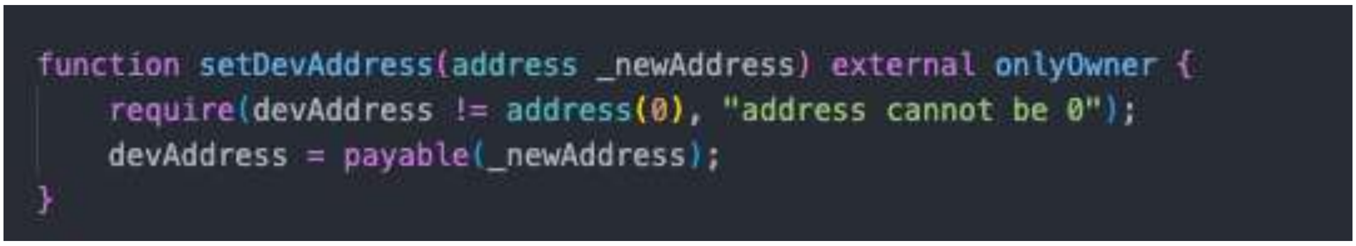
```
function renounceOwnership() external onlyOwner {
    address oldOwner = _owner;
    _owner = address(0);
    emit OwnershipTransferred(oldOwner, address(0));
}
```

❖ The owner can exclude pre-sale address before adding lps



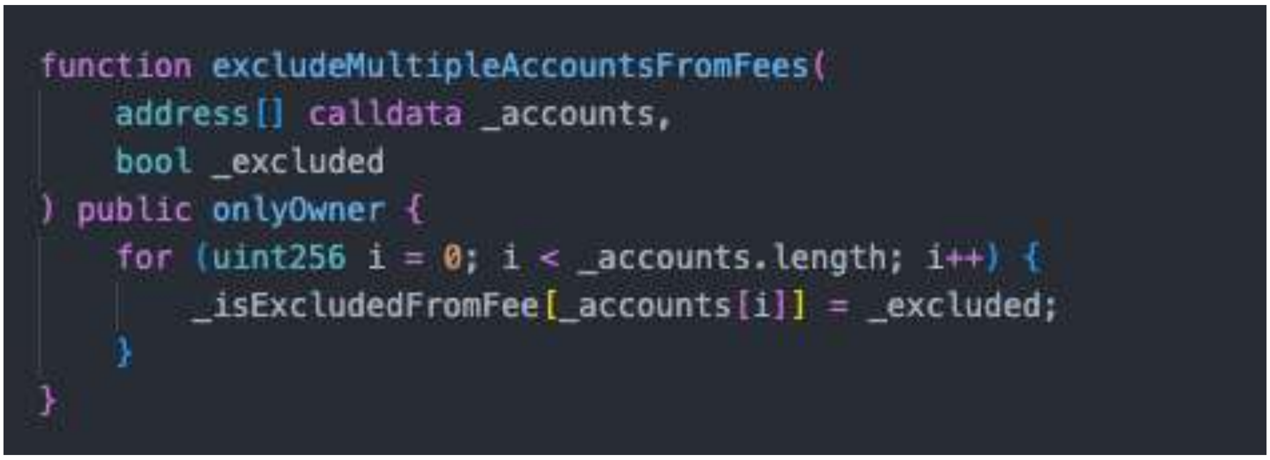
```
function excludePresaleAddresses(
    address _router,
    address _presale
) external onlyOwner {
    require(allowedPresaleExclusion);
    require(
        _router != address(this) &&
            _presale != address(this) &&
            lpPair != _router &&
            lpPair != _presale,
        "Just don't."
    );
    if (_router == _presale) {
        _liquidityHolders[_presale] = true;
    } else {
        _liquidityHolders[_router] = true;
        _liquidityHolders[_presale] = true;
    }
}
```

❖ The owner can change the dev wallet address



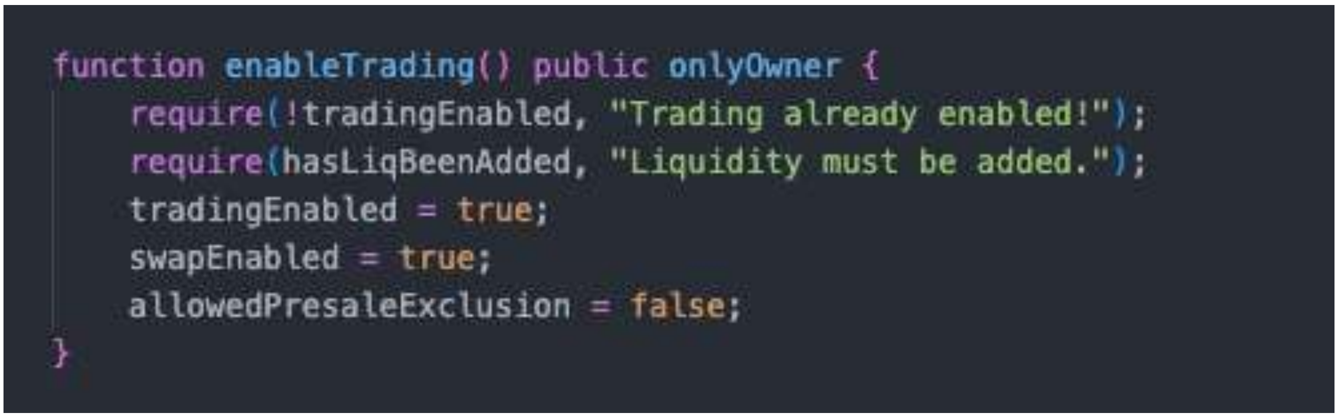
```
function setDevAddress(address _newAddress) external onlyOwner {
    require(devAddress != address(0), "address cannot be 0");
    devAddress = payable(_newAddress);
}
```

❖ The owner include/exclude wallets from fees



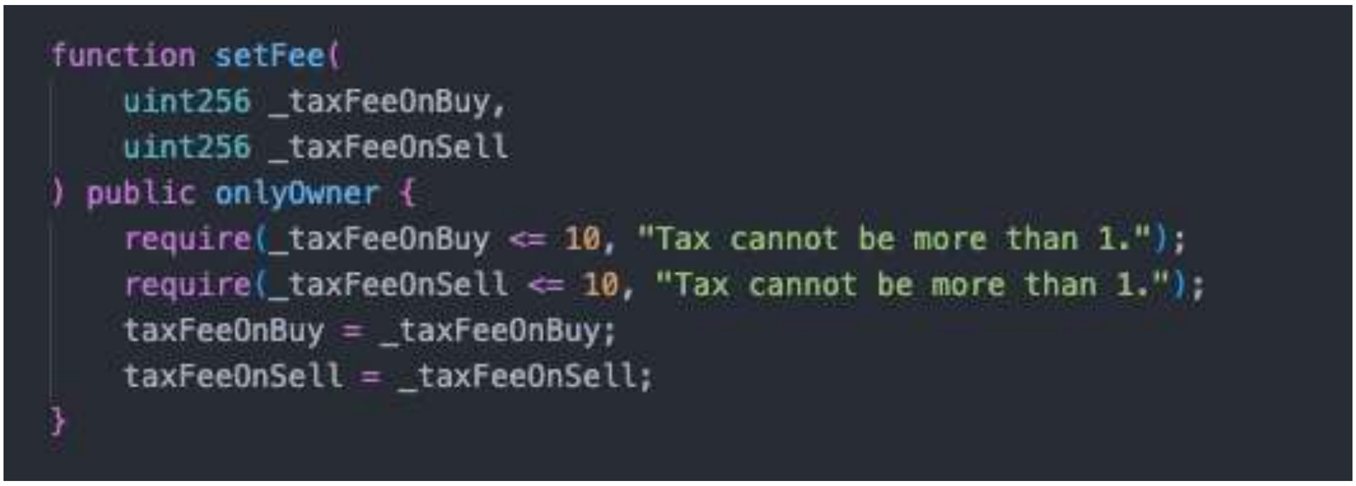
```
function excludeMultipleAccountsFromFees(
    address[] calldata _accounts,
    bool _excluded
) public onlyOwner {
    for (uint256 i = 0; i < _accounts.length; i++) {
        _isExcludedFromFee[_accounts[i]] = _excluded;
    }
}
```

❖ The owner can enable trading, but once enabled can not disable it again



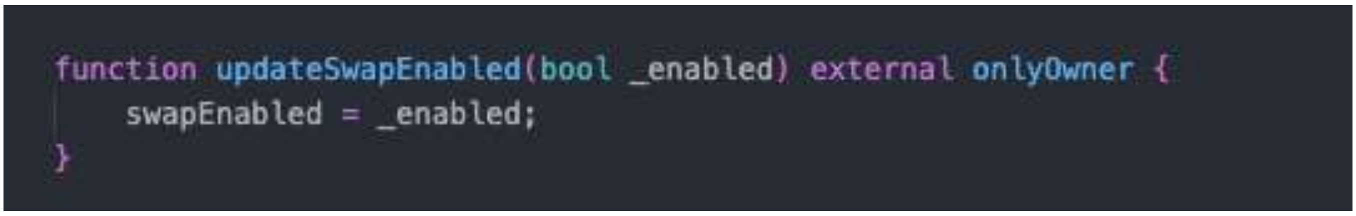
```
function enableTrading() public onlyOwner {
    require(!tradingEnabled, "Trading already enabled!");
    require(hasLiqBeenAdded, "Liquidity must be added.");
    tradingEnabled = true;
    swapEnabled = true;
    allowedPresaleExclusion = false;
}
```

❖ The owner can change sell and buy fees for maximum upto 10% of each fee



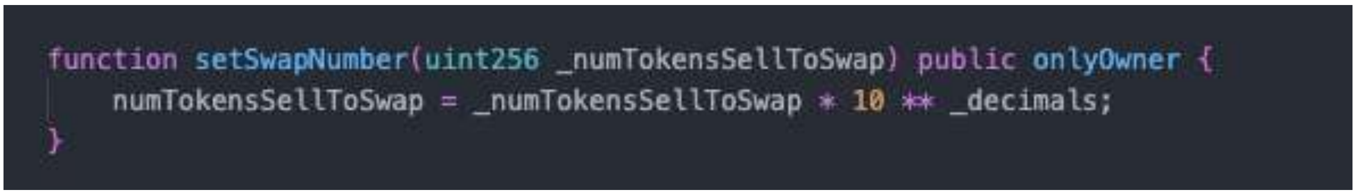
```
function setFee(
    uint256 _taxFeeOnBuy,
    uint256 _taxFeeOnSell
) public onlyOwner {
    require(_taxFeeOnBuy <= 10, "Tax cannot be more than 1.");
    require(_taxFeeOnSell <= 10, "Tax cannot be more than 1.");
    taxFeeOnBuy = _taxFeeOnBuy;
    taxFeeOnSell = _taxFeeOnSell;
}
```

❖ The owner can enable/disable swapping



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

❖ The owner can change the swap point



```
function setSwapNumber(uint256 _numTokensSellToSwap) public onlyOwner {
    numTokensSellToSwap = _numTokensSellToSwap * 10 ** _decimals;
}
```

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: **2**

Solidity code functional issue level: **PASS**

Number of owner privileges: **12**

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

Smart contract active ownership: **ACTIVE**