# RugFreeCoins Audit

SUNSHIELD

## Sunshield Token

## Smart Contract Security Audit

April 30, 2022

# Contents

# Audit details

**Audited project**
SunShield Token

**Contract Address**
0x8bb595c140c60d833ccb6c814f8a19b5a2541615

**Client contact**
SunShield Team

**Blockchain**
Binance smart chain

**Project website**
https://www.sunshield.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 SunShield V2 Token to perform an audit of the smart contract.

**https://bscscan.com/address/0x8bb595c140c60d833ccb6c814f8a19b5a2541615**

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

SunShield Token is a token built on the Binance Smart Chain. SSV2 gives holders the ability to choose what token they get as a reward BEP20 only and ran through the pancake router. SSV2 also allows you to split your reward into percentages and claim BNB as well as another token. For example, holders can choose to receive (50% BUSD/ 50% BNB) as a reward or replace BUSD with whatever they like. SSV2 is part of a bigger utility which will be our ecosystem that is already developed and in the beta testing phase. This includes a staking platform, Cross-chain DEX & NFT marketplace, Launchpad, and Metaverse Each transaction, purchase, and sale incur a 15% fee.

The **automatic 50% BNB/ 50% BUSD or rewards in any other token instead BUSDs that holders prefer in a 4% fee when buying and selling** is what SunShield V2's unique marketing strategy is based around: that BNB and other rewards will be distributed among every holder proportional to how many tokens each individual holds.

- The liquidity fee of 2%, is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.

- A fee of 1% when buying and selling are charged for Bnb buys back from the supply and burns all tokens bought automatically.

- The **sustainability fee of 7% marketing** is what allows SunShield V2 to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to a marketing wallet. This way, SunShield will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

- The fee of 1% will be deducted and swapped into BNBs and sent to the team wallet for salaries.

# ROADMAP

**PHASE 01**

- LAUNCH ON FEG SMART DEFI
- PRESALE / V1 HOLDER AIRDROP
- LAUNCH ON PCS
- 10,000 HOLDERS
- LOGO ON TRUST WALLET
- AGGRESSIVE MARKETING CAMPAIGN
- P2E GAME DEVELOPMENT
- SUNSHIELD 3D NFTS
- SUNSHIELD TREASURY
- SUNSHIELD DAPP
- 10,000 TWITTER FOLLOWERS
- 10,000 INSTAGRAM FOLLOWERS

**PHASE 02**

- TECH RATE AUDIT
- CERTIK AUDIT
- STAKING
- NFT STAKING
- DAPP UPGRADE
- MIGRATE SUNSHIELD LP TO ESHIELD LOCKER
- P2E GAME WHITE PAPER
- P2E GAME BETA
- SUNSHIELD CHARITY PROGRAM
- SUNSHIELD RELEASE ON CRONOS, POLYGON, AVAX, ETH, CARDANO

**PHASE 03**

- SUNSHIELD DAO
- SUNSHIELD SITE UPGRADE
- SUNSHIELD NFT COLLECTION #2
- P2E GAME COLLECTIBLES
- P2E GAME RELEASE

## Tokenomics

**15% fee when buying and selling**

- 4% of trade goes to holders pockets in BNB/BUSD or any other token
- 2% of trade goes to the liquidity pool.
- 7% of trade goes to marketing.
- 1% of trade goes to buyback & burn.
- 1% of trade goes to the Team.

# 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/BUSD or any other token they prefer by holding tokens.
- Anyone who’s interested in trading tokens.
- Anyone who’s interested in collecting and trading NFTs
- Anyone who’s interested in staking tokens and getting rewards.
- Anyone who’s interested in taking part in the future plans of the SunShield project.
- Anyone who’s interested in making financial transactions with any other party using SunShield tokens or BNB/BUSD or other tokens gets rewards through the platform as the currency.

## Core concept

### The reward system

**4% of each transaction when buying and selling** allows holders to split rewards into percentages and claim BNB as well as another token. For example, holders can choose to receive (50% BUSD/ 50% BNB) as a reward or replace BUSD with whatever they likeBNB, and is split amongst all holders. The rewards are sent to holders that have at least 777.777 SunShield tokens, holders will be eligible to receive tokens every 12 hours, and rewards are proportional to how many tokens each individual holds.

### Sustainable mechanism

**The liquidity fee of 1%,** which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

**The fee of 7% when buying and selling for marketing and dev** is what allows SunShield V2 to promote the token and use funds to further the development of the platform. Tokens will be swapped into BNB/BUSD or any other tokens and will be sent to a marketing wallet. This way, SunShield V2 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 fee of 1% when buying and selling** for the team is getting swapped to BNBs and sent to the team wallet.

**The buyback and burn mechanism collect a 1% tax when buying and selling**, which is stored inside the contract. Whenever a buy or sell occurs, a fraction of the buyback amount is used to automatically purchase tokens from the liquidity pool. Those tokens are immediately burned after purchase, which keeps the token price stable.

# Potential to grow with score points

| | | |
|---|---|---|
| 1. | Project efficiency | 9/10 |
| 2. | Project uniqueness | 10/10 |
| 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 | | **9/10** |

# Contract details

## Token contract details for 30th April 2022

| | |
|---|---|
| Contract name | Sunshield V2 |
| Contract address | 0x8bB595C140c60D833CcB6c814F8A19B5A2541615 |
| Token supply | 777,777 |
| Token ticker | SSV2 |
| Decimals | 9 |
| Token holders | 174 |
| Transaction count | 595 |
| Burn wallet | 0x000000000000000000000000000000000000dead |
| Auto liquidity wallet | 0x2cd135f3ebf1c5082ac88eb658ed5e8e12d7c2cd |
| Dev Marketing wallet | 0xb0f8dd8469dcb6e1eb50a818fe206f0d1299936e |
| Get Salary wallet | 0x3d4e145649179d54093065f9125f9e78087ec5ee |
| Contract deployer address | 0x2Cd135f3ebf1C5082aC88eb658Ed5E8e12d7c2cd |
| Contract’s current owner address | 0x2cd135f3ebf1c5082ac88eb658ed5e8e12d7c2cd |

# 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 | High centralization risk |
| | | 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❗ |
| | | | | |
| **IERC20Metadata** | **Interface** | **IERC20** | | |
| └ | name | External ❗ | | NO❗ |
| └ | symbol | External ❗ | | NO❗ |
| └ | decimals | External ❗ | | NO❗ |
| | | | | |
| **Context** | **Implementation** | | | |
| └ | _msgSender | Internal 🔒 | | |
| └ | _msgData | Internal 🔒 | | |

| | | | | |
|---|---|---|---|---|
| | | | | |
| **Ownable** | **Implementation** | **Context** | | |
| └ | | Public ❗ | 🛑 | NO❗ |
| └ | owner | Public ❗ | | NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑 | onlyOwner |
| | | | | |
| **IPancakeRouter01** | **Interface** | | | |
| └ | factory | External ❗ | | NO❗ |
| └ | WETH | External ❗ | | NO❗ |
| └ | addLiquidity | External ❗ | 🛑 | NO❗ |
| └ | addLiquidityETH | External ❗ | 💵 | NO❗ |
| └ | removeLiquidity | External ❗ | 🛑 | NO❗ |
| └ | removeLiquidityETH | External ❗ | 🛑 | NO❗ |
| └ | removeLiquidityWithPermit | External ❗ | 🛑 | NO❗ |
| └ | removeLiquidityETHWithPermit | External ❗ | 🛑 | NO❗ |
| └ | swapExactTokensForTokens | External ❗ | 🛑 | NO❗ |
| └ | swapTokensForExactTokens | External ❗ | 🛑 | NO❗ |
| └ | swapExactETHForTokens | External ❗ | 💵 | NO❗ |
| └ | swapTokensForExactETH | External ❗ | 🛑 | NO❗ |
| └ | swapExactTokensForETH | External ❗ | 🛑 | NO❗ |
| └ | swapETHForExactTokens | External ❗ | 💵 | NO❗ |
| └ | quote | External ❗ | | NO❗ |

|  |  |  |  |  |
|---|---|---|---|---|
| └ | getAmountOut | External ❗ |  | NO❗ |
| └ | getAmountIn | External ❗ |  | NO❗ |
| └ | getAmountsOut | External ❗ |  | NO❗ |
| └ | getAmountsIn | External ❗ |  | NO❗ |
|  |  |  |  |  |
| **IPancakeRouter02** | **Interface** | **IPancake Router01** |  |  |
| └ | removeLiquidityETHSupportingFeeOnTransferTokens | External ❗ | 🛑 | NO❗ |
| └ | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External ❗ | 🛑 | NO❗ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | 🛑 | NO❗ |
| └ | swapExactETHForTokensSupportingFeeOnTransferTokens | External ❗ | 💵 | NO❗ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗ | 🛑 | NO❗ |
|  |  |  |  |  |
| **IPancakeFactory** | **Interface** |  |  |  |
| └ | feeTo | External ❗ |  | NO❗ |
| └ | feeToSetter | External ❗ |  | NO❗ |
| └ | getPair | External ❗ |  | NO❗ |
| └ | allPairs | External ❗ |  | NO❗ |
| └ | allPairsLength | External ❗ |  | NO❗ |
| └ | createPair | External ❗ | 🛑 | NO❗ |
| └ | setFeeTo | External ❗ | 🛑 | NO❗ |
| └ | setFeeToSetter | External ❗ | 🛑 | NO❗ |

| | | | | |
|---|---|---|---|---|
| | | | | |
| **Reentrancy Guard** | **Implementation** | | | |
| L | | Public ❗ | 🛑 | NO❗ |
| | | | | |
| **IPinkAntiBot** | **Interface** | | | |
| L | setTokenOwner | External ❗ | 🛑 | NO❗ |
| L | onPreTransferCheck | External ❗ | 🛑 | NO❗ |
| | | | | |
| **SSV2Base** | **Implementation** | **Context, IERC20Metadata, Ownable, Reentrancy Guard** | | |
| L | | Public ❗ | 🛑 | NO❗ |
| L | setEnableAntiBot | External ❗ | 🛑 | onlyOwner |
| L | activate | Public ❗ | 🛑 | onlyOwner |
| L | onActivated | Internal 🔒 | 🛑 | |
| L | balanceOf | Public ❗ | | NO❗ |
| L | transfer | Public ❗ | 🛑 | NO❗ |
| L | transferFrom | Public ❗ | 🛑 | NO❗ |
| L | approve | Public ❗ | 🛑 | NO❗ |
| L | doTransfer | Internal 🔒 | 🛑 | |
| L | onBeforeTransfer | Internal 🔒 | 🛑 | |
| L | onTransfer | Internal 🔒 | 🛑 | |

| | | | | |
|---|---|---|---|---|
| └ | updateBalances | Private 🔐 | 🛑 | |
| └ | doApprove | Private 🔐 | 🛑 | |
| └ | calculateFeeRate | Private 🔐 | | |
| └ | onBeforeCalculateFeeRate | Internal 🔒 | | |
| └ | executeSwapIfNeeded | Private 🔐 | 🛑 | |
| └ | executeSwap | Private 🔐 | 🛑 | |
| └ | swapTokensForBNB | Internal 🔒 | 🛑 | |
| └ | swapBNBForTokens | Internal 🔒 | 🛑 | |
| └ | swapBNBForCustomeTokens | Internal 🔒 | 🛑 | |
| └ | isTransferLimited | Private 🔐 | | |
| └ | isSwapTransfer | Private 🔐 | | |
| └ | isMarketTransfer | Internal 🔒 | | |
| └ | amountUntilSwap | Public ❗ | | NO❗ |
| └ | increaseAllowance | Public ❗ | 🛑 | NO❗ |
| └ | decreaseAllowance | Public ❗ | 🛑 | NO❗ |
| └ | setPancakeSwapRouter | Public ❗ | 🛑 | onlyOwner |
| └ | onPancakeSwapRouterUpdated | Internal 🔒 | 🛑 | |
| └ | isPancakeSwapPair | Internal 🔒 | | |
| └ | setBuyFees | Public ❗ | 🛑 | onlyOwner |
| └ | setSellFees | Public ❗ | 🛑 | onlyOwner |
| └ | adjustTaxes | Private 🔐 | 🛑 | |

|  |  |  |  |  |
|---|---|---|---|---|
| └ | setTransactionLimit | Public ❗ | 🛑 | onlyOwner |
| └ | transactionLimit | Public ❗ |  | NO❗ |
| └ | setTokenSwapThreshold | Public ❗ | 🛑 | onlyOwner |
| └ | tokenSwapThreshold | Public ❗ |  | NO❗ |
| └ | name | Public ❗ |  | NO❗ |
| └ | symbol | Public ❗ |  | NO❗ |
| └ | totalSupply | Public ❗ |  | NO❗ |
| └ | decimals | Public ❗ |  | NO❗ |
| └ | allowance | Public ❗ |  | NO❗ |
| └ | pancakeSwapRouterAddress | Public ❗ |  | NO❗ |
| └ | pancakeSwapPairAddress | Public ❗ |  | NO❗ |
| └ | autoLiquidityWallet | Public ❗ |  | NO❗ |
| └ | setAutoLiquidityWallet | Public ❗ | 🛑 | onlyOwner |
| └ | devmarketingWallet | Public ❗ |  | NO❗ |
| └ | setMarketingWallet | Public ❗ | 🛑 | onlyOwner |
| └ | getSalaryWallet | Public ❗ |  | NO❗ |
| └ | setSalaryWallet | Public ❗ | 🛑 | onlyOwner |
| └ | totalFeesPooled | Public ❗ |  | NO❗ |
| └ | totalBNBLiquidityAddedFromFees | Public ❗ |  | NO❗ |
| └ | isSwapEnabled | Public ❗ |  | NO❗ |
| └ | setSwapEnabled | Public ❗ | 🛑 | onlyOwner |

|  |  |  |  |  |
|---|---|---|---|---|
| └ | isFeeEnabled | Public ❗ |  | NO❗ |
| └ | setFeeEnabled | Public ❗ | 🛑 | onlyOwner |
| └ | isExcludedFromFees | Public ❗ |  | NO❗ |
| └ | setExcludedFromFees | Public ❗ | 🛑 | onlyOwner |
| └ | activateBuying | Public ❗ | 🛑 | onlyOwner |
| └ |  | External ❗ | 💵 | NO❗ |
|  |  |  |  |  |
| **SSV2** | **Implementation** | **SSV2Base** |  |  |
| └ |  | Public ❗ | 🛑 | SSV2Base |
| └ | onActivated | Internal 🔒 | 🛑 |  |
| └ | onBeforeTransfer | Internal 🔒 | 🛑 |  |
| └ | onTransfer | Internal 🔒 | 🛑 |  |
| └ | processGradualBurn | Private 🔐 | 🛑 |  |
| └ | updateAutoClaimQueue | Private 🔐 | 🛑 |  |
| └ | doClaimReward | Private 🔐 | 🛑 |  |
| └ | claimBNB | Private 🔐 | 🛑 |  |
| └ | claimRewardToken | Private 🔐 | 🛑 |  |
| └ | processRewardClaimQueue | Public ❗ | 🛑 | NO❗ |
| └ | processRewardClaimQueueAndRefund Gas | External ❗ | 🛑 | NO❗ |
| └ | isRewardReady | Public ❗ |  | NO❗ |
| └ | isIncludedInRewards | Public ❗ |  | NO❗ |

| | | | | |
|---|---|---|---|---|
| └ | calculateRewardCycleExtension | Public ❗ | | NO❗ |
| └ | calculateClaimRewards | Public ❗ | | NO❗ |
| └ | calculateBNBReward | Public ❗ | | NO❗ |
| └ | onPancakeSwapRouterUpdated | Internal 🔒 | 🛑 | |
| └ | isMarketTransfer | Internal 🔒 | | |
| └ | isBurnTransfer | Private 🔐 | | |
| └ | shouldBurn | Public ❗ | | NO❗ |
| └ | buyAndBurn | External ❗ | 🛑 | onlyOwner |
| └ | doBuyAndBurn | Private 🔐 | 🛑 | |
| └ | isContract | Public ❗ | | NO❗ |
| └ | totalAmountOfTokensHeld | Public ❗ | | NO❗ |
| └ | bnbRewardClaimed | Public ❗ | | NO❗ |
| └ | bnbRewardClaimedAsSSV2 | Public ❗ | | NO❗ |
| └ | totalBNBClaimed | Public ❗ | | NO❗ |
| └ | totalBNBClaimedAsSSV2 | Public ❗ | | NO❗ |
| └ | rewardCyclePeriod | Public ❗ | | NO❗ |
| └ | setRewardCyclePeriod | Public ❗ | 🛑 | onlyOwner |
| └ | setRewardCycleExtensionThreshold | Public ❗ | 🛑 | onlyOwner |
| └ | nextAvailableClaimDate | Public ❗ | | NO❗ |
| └ | maxClaimAllowed | Public ❗ | | NO❗ |
| └ | setMaxClaimAllowed | Public ❗ | 🛑 | onlyOwner |

| | | | | |
|---|---|---|---|---|
| └ | minRewardBalance | Public ❗️ | | NO❗️ |
| └ | setMinRewardBalance | Public ❗️ | 🛑 | onlyOwner |
| └ | maxGasForAutoClaim | Public ❗️ | | NO❗️ |
| └ | setMaxGasForAutoClaim | Public ❗️ | 🛑 | onlyOwner |
| └ | isAutoClaimEnabled | Public ❗️ | | NO❗️ |
| └ | setAutoClaimEnabled | Public ❗️ | 🛑 | onlyOwner |
| └ | isExcludedFromRewards | Public ❗️ | | NO❗️ |
| └ | setExcludedFromRewards | Public ❗️ | 🛑 | onlyOwner |
| └ | globalRewardDampeningPercentage | Public ❗️ | | NO❗️ |
| └ | setGlobalRewardDampeningPercentage | Public ❗️ | 🛑 | onlyOwner |
| └ | approveClaim | Public ❗️ | 🛑 | NO❗️ |
| └ | isClaimApproved | Public ❗️ | | NO❗️ |
| └ | isRewardAsTokensEnabled | Public ❗️ | | NO❗️ |
| └ | setRewardAsTokensEnabled | Public ❗️ | 🛑 | onlyOwner |
| └ | gradualBurnMagnitude | Public ❗️ | | NO❗️ |
| └ | setGradualBurnMagnitude | Public ❗️ | 🛑 | onlyOwner |
| └ | gradualBurnTimespan | Public ❗️ | | NO❗️ |
| └ | setGradualBurnTimespan | Public ❗️ | 🛑 | onlyOwner |
| └ | claimRewardAsTokensPercentage | Public ❗️ | | NO❗️ |
| └ | setClaimRewardAsTokensPercentage | Public ❗️ | 🛑 | NO❗️ |
| └ | mainBnbPoolSize | Public ❗️ | | NO❗️ |

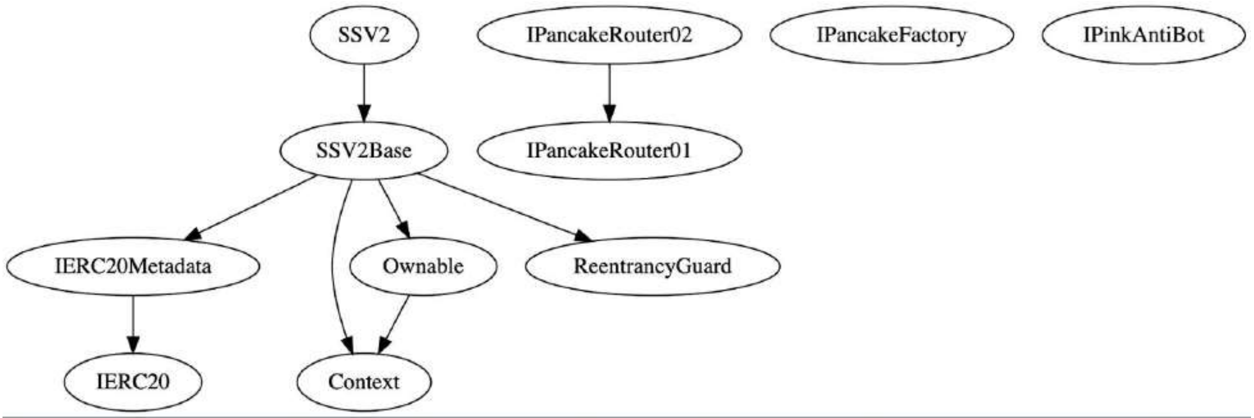
| | | | | |
|---|---|---|---|---|
| └ | setMainBnbPoolSize | Public ❗ | 🛑 | onlyOwner |
| └ | isInRewardClaimQueue | Public ❗ | | NO❗ |
| └ | reimburseAfterSSV2ClaimFailure | Public ❗ | | NO❗ |
| └ | setReimburseAfterSSV2ClaimFailure | Public ❗ | 🛑 | onlyOwner |
| └ | lastBurnDate | Public ❗ | | NO❗ |
| └ | rewardClaimQueueLength | Public ❗ | | NO❗ |
| └ | rewardClaimQueueIndex | Public ❗ | | NO❗ |
| └ | isWhitelistedExternalProcessor | Public ❗ | | NO❗ |
| └ | setWhitelistedExternalProcessor | Public ❗ | 🛑 | onlyOwner |
| └ | setSendWeiGasLimit | Public ❗ | 🛑 | onlyOwner |
| └ | setExcludeNonHumansFromRewards | Public ❗ | 🛑 | onlyOwner |
| └ | blacklistAccount | Public ❗ | 🛑 | onlyOwner |
| └ | unBlacklistAccount | Public ❗ | 🛑 | onlyOwner |
| └ | claimRewardAsToken | Public ❗ | | NO❗ |
| └ | setClaimRewardAsToken | Public ❗ | 🛑 | NO❗ |
| └ | setClaimRewardAsTokenAndPercentag e | Public ❗ | 🛑 | NO❗ |
| └ | _getNow | Private 🔐 | | |
| └ | activateTrading | Public ❗ | 🛑 | onlyOwner |
| └ | deactivateTrading | Public ❗ | 🛑 | onlyOwner |
| └ | setHappyHour | Public ❗ | 🛑 | onlyOwner |

| | | | | |
|---|---|---|---|---|
| └ | cancelHappyHour | Public ❗️ | 🛑 | onlyOwner |
| └ | isInHappyHour | External ❗️ | | NO❗️ |
| └ | setHappyHour1SellFees | Public ❗️ | 🛑 | onlyOwner |
| └ | setHappyHour2SellFees | Public ❗️ | 🛑 | onlyOwner |
| └ | setHappyHour1BuyFees | Public ❗️ | 🛑 | onlyOwner |
| └ | setHappyHour2BuyFees | Public ❗️ | 🛑 | onlyOwner |
| └ | _setCustomSellTaxPeriod | Private 🔐 | 🛑 | |
| └ | _setCustomBuyTaxPeriod | Private 🔐 | 🛑 | |
| └ | getHappyHour1BuyFees | External ❗️ | | NO❗️ |
| └ | getHappyHour1SellFees | External ❗️ | | NO❗️ |
| └ | getHappyHour2BuyFees | External ❗️ | | NO❗️ |
| └ | getHappyHour2SellFees | External ❗️ | | NO❗️ |
| └ | _isInHappyHour1 | Internal 🔒 | | |
| └ | _isInHappyHour2 | Internal 🔒 | | |
| └ | onBeforeCalculateFeeRate | Internal 🔒 | | |

**Legend**

| Symbol | Meaning |
|---|---|
| 🛑 | Function can modify state |
| 💵 | Function is payable |

## Inheritance Hierarchy



# Security issue checking status

❖ **High severity issues**
No High severity issues found

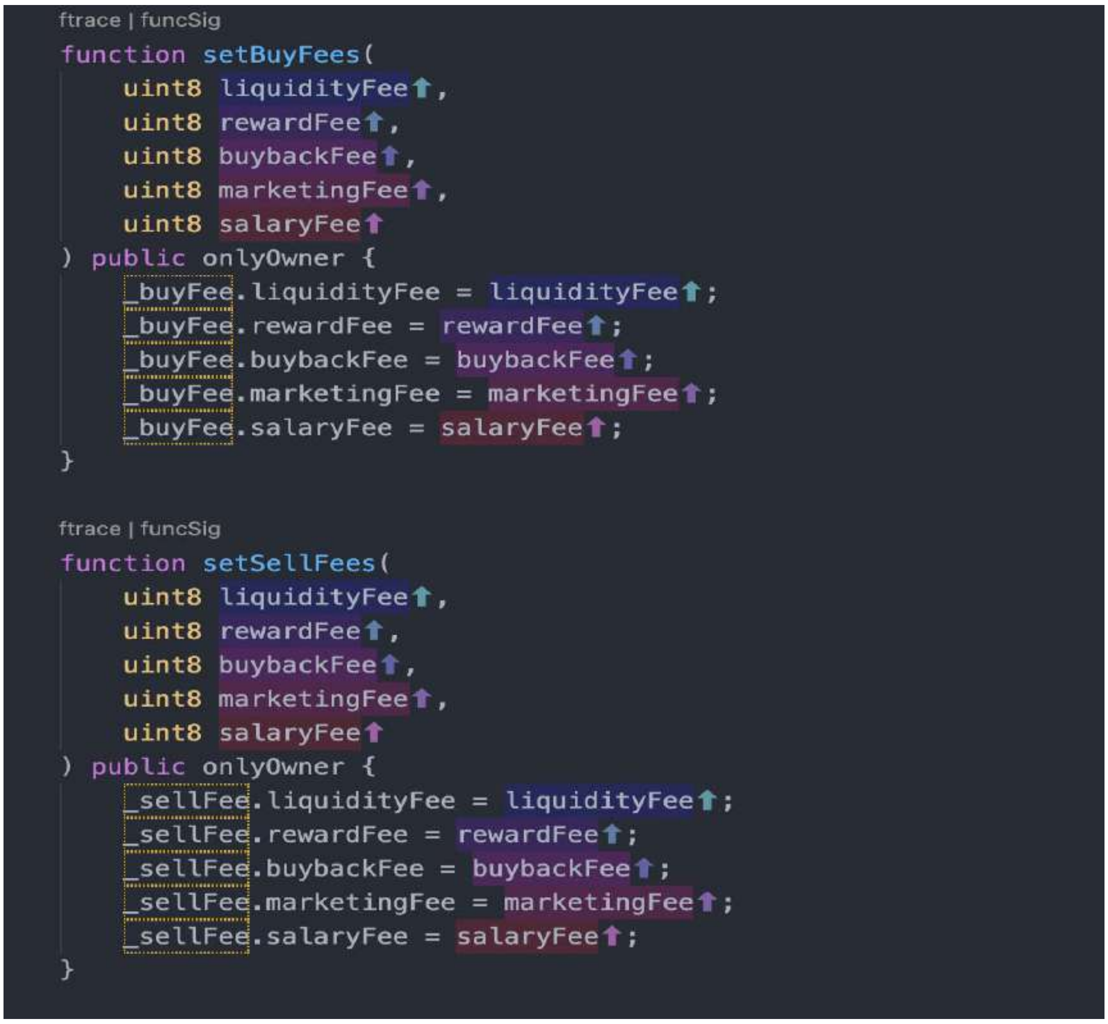
❖ **Medium severity issues**
No medium severity issues found

❖ **Low severity issues**
No low severity issues found

❖ **Centralization Risk**

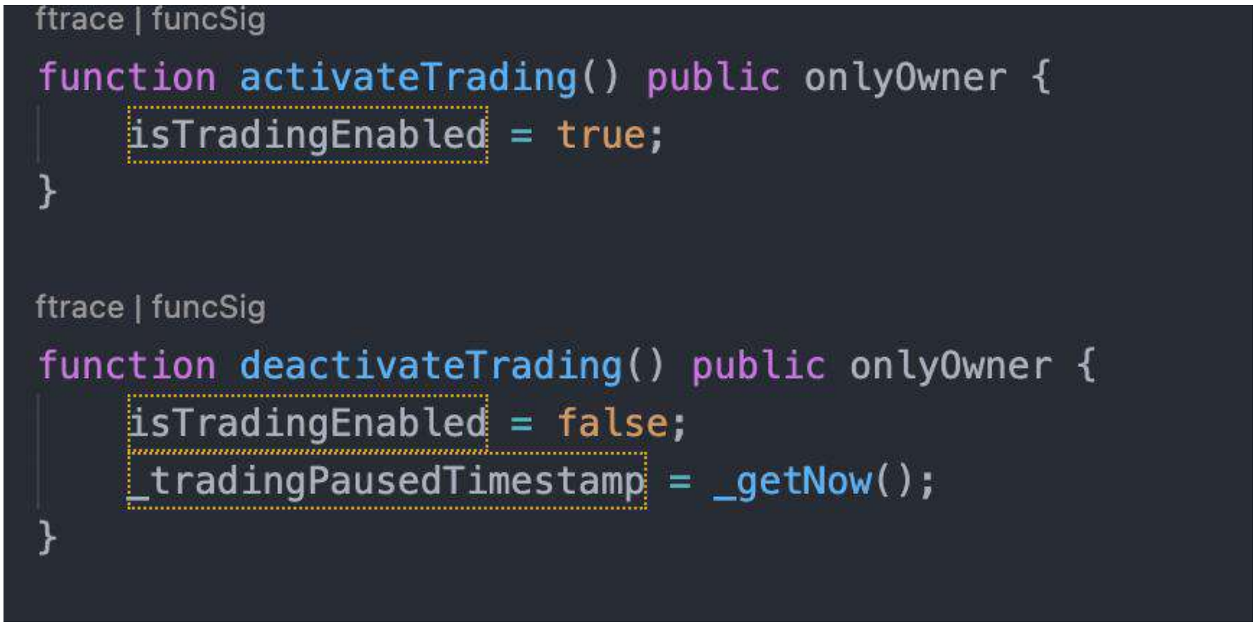
High Centralization Risks

❖ The owner can change all buy and sell fees without any maximum limit (can set to 100%)



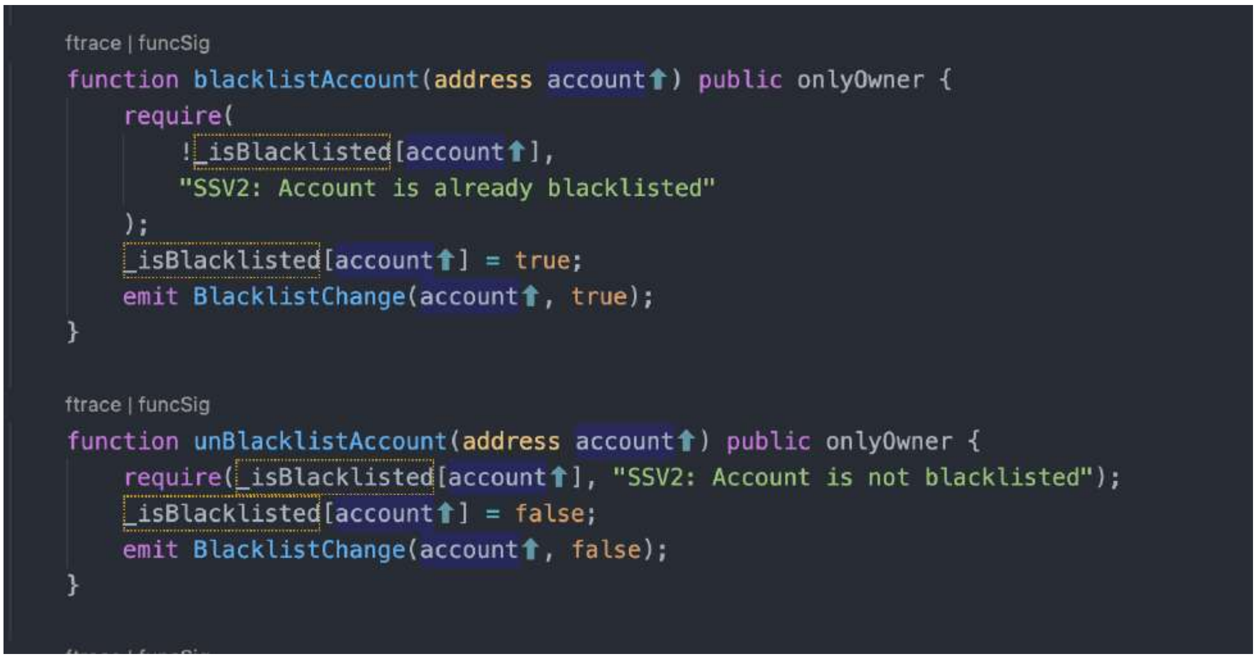
```
ftrace | funcSig
function setBuyFees(
    uint8 liquidityFee,
    uint8 rewardFee,
    uint8 buybackFee,
    uint8 marketingFee,
    uint8 salaryFee
) public onlyOwner {
    _buyFee.liquidityFee = liquidityFee;
    _buyFee.rewardFee = rewardFee;
    _buyFee.buybackFee = buybackFee;
    _buyFee.marketingFee = marketingFee;
    _buyFee.salaryFee = salaryFee;
}

ftrace | funcSig
function setSellFees(
    uint8 liquidityFee,
    uint8 rewardFee,
    uint8 buybackFee,
    uint8 marketingFee,
    uint8 salaryFee
) public onlyOwner {
    _sellFee.liquidityFee = liquidityFee;
    _sellFee.rewardFee = rewardFee;
    _sellFee.buybackFee = buybackFee;
    _sellFee.marketingFee = marketingFee;
    _sellFee.salaryFee = salaryFee;
}
```

❖ The owner can active and deactivate trading any time



```
ftrace | funcSig
function activateTrading() public onlyOwner {
    isTradingEnabled = true;
}

ftrace | funcSig
function deactivateTrading() public onlyOwner {
    isTradingEnabled = false;
    _tradingPausedTimestamp = _getNow();
}
```

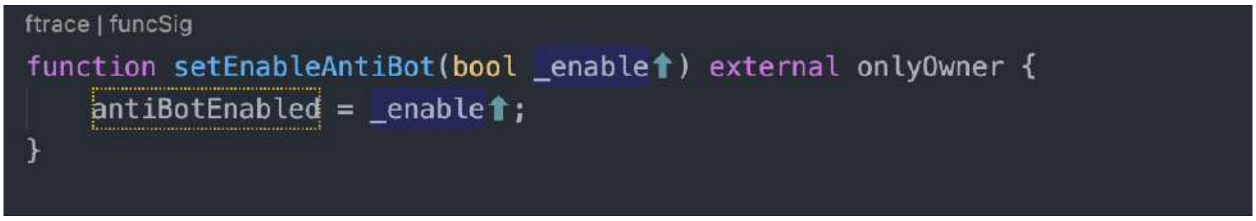
❖ The owner can blacklist and unblack list wallets



```
ftrace | funcSig
function blacklistAccount(address account) public onlyOwner {
    require(
        !_isBlacklisted[account],
        "SSV2: Account is already blacklisted"
    );
    _isBlacklisted[account] = true;
    emit BlacklistChange(account, true);
}

ftrace | funcSig
function unBlacklistAccount(address account) public onlyOwner {
    require(_isBlacklisted[account], "SSV2: Account is not blacklisted");
    _isBlacklisted[account] = false;
    emit BlacklistChange(account, false);
}
```

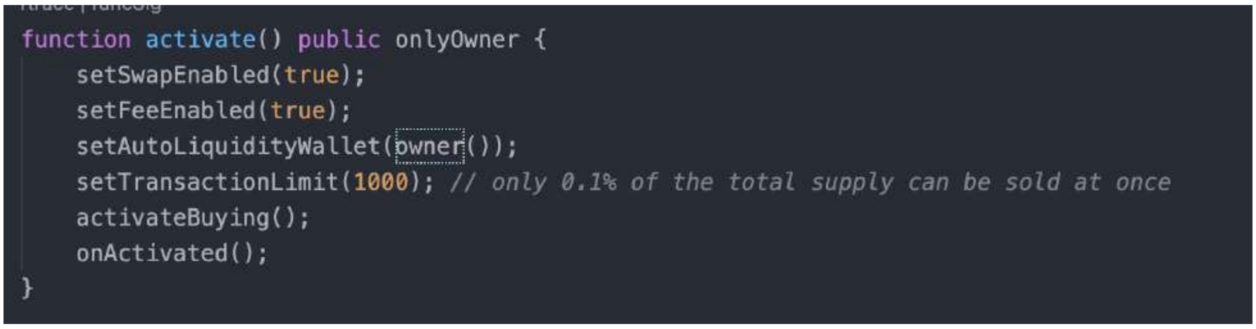
# Owner privileges

❖ The owner can enable/disable pink antibot



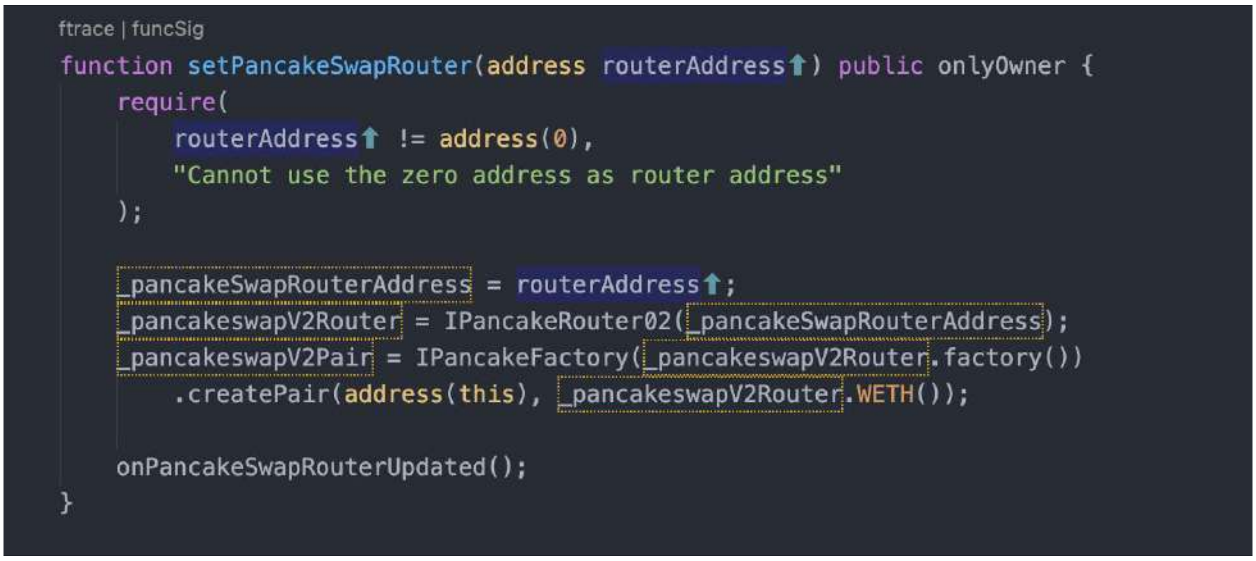
```
ftrace | funcSig
function setEnableAntiBot(bool _enable) external onlyOwner {
    antiBotEnabled = _enable;
}
```

❖ The owner can activate the contract



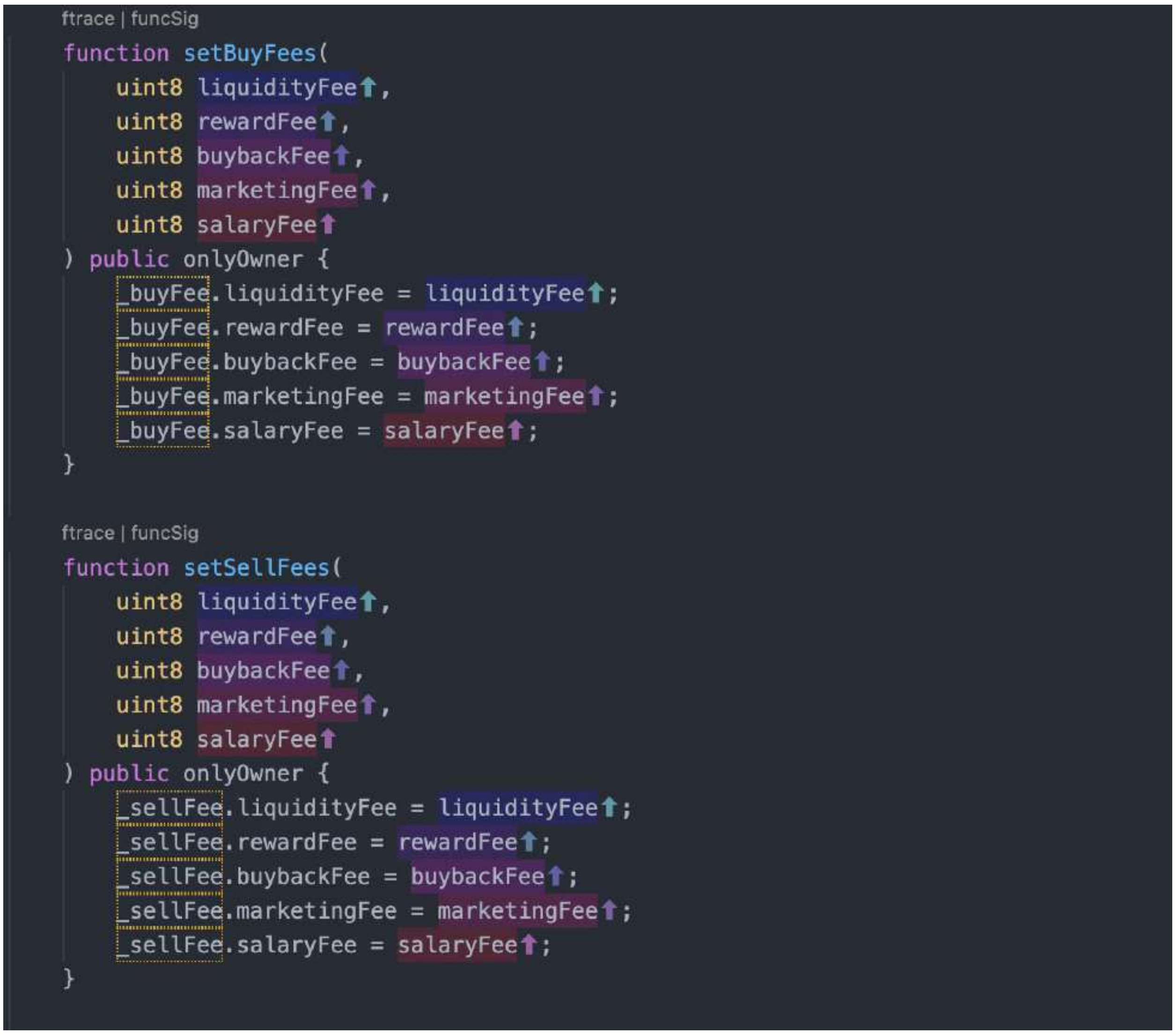
```
function activate() public onlyOwner {
    setSwapEnabled(true);
    setFeeEnabled(true);
    setAutoLiquidityWallet(owner());
    setTransactionLimit(1000); // only 0.1% of the total supply can be sold at once
    activateBuying();
    onActivated();
}
```

❖ The owner can change the router address



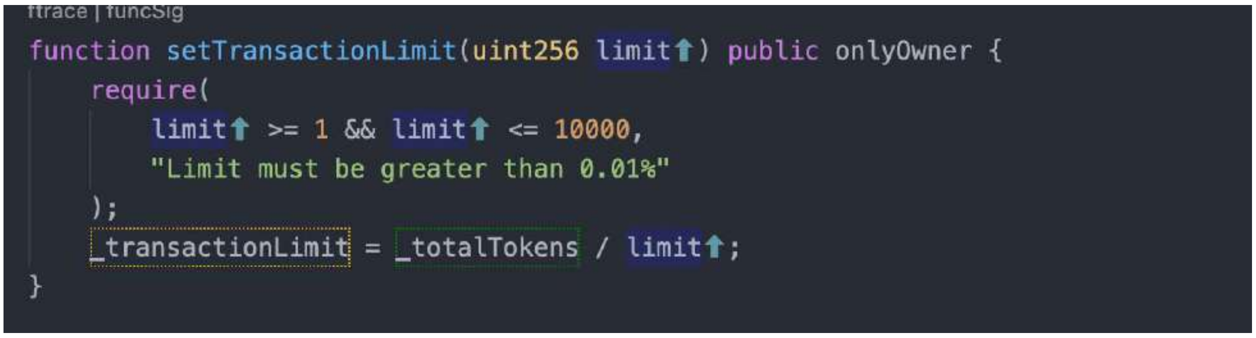
```
ftrace | funcSig
function setPancakeSwapRouter(address routerAddress) public onlyOwner {
    require(
        routerAddress != address(0),
        "Cannot use the zero address as router address"
    );

    _pancakeSwapRouterAddress = routerAddress;
    _pancakeswapV2Router = IPancakeRouter02(_pancakeSwapRouterAddress);
    _pancakeswapV2Pair = IPancakeFactory(_pancakeswapV2Router.factory())
        .createPair(address(this), _pancakeswapV2Router.WETH());

    onPancakeSwapRouterUpdated();
}
```

❖ The owner can change all buy and sell fees



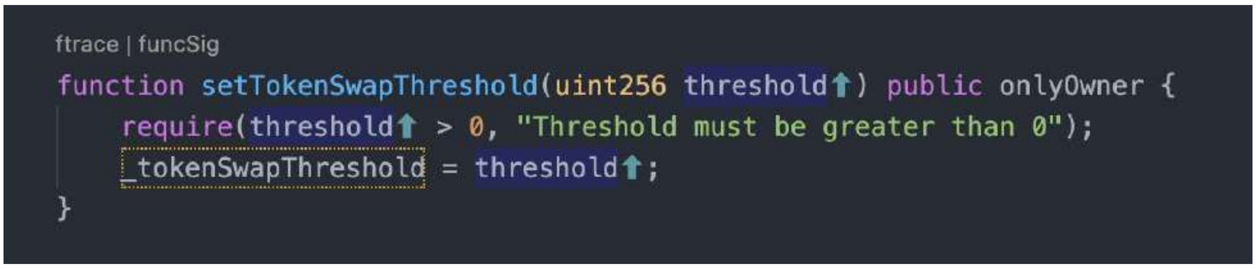
```
ftrace | funcSig
function setBuyFees(
    uint8 liquidityFee,
    uint8 rewardFee,
    uint8 buybackFee,
    uint8 marketingFee,
    uint8 salaryFee
) public onlyOwner {
    _buyFee.liquidityFee = liquidityFee;
    _buyFee.rewardFee = rewardFee;
    _buyFee.buybackFee = buybackFee;
    _buyFee.marketingFee = marketingFee;
    _buyFee.salaryFee = salaryFee;
}

ftrace | funcSig
function setSellFees(
    uint8 liquidityFee,
    uint8 rewardFee,
    uint8 buybackFee,
    uint8 marketingFee,
    uint8 salaryFee
) public onlyOwner {
    _sellFee.liquidityFee = liquidityFee;
    _sellFee.rewardFee = rewardFee;
    _sellFee.buybackFee = buybackFee;
    _sellFee.marketingFee = marketingFee;
    _sellFee.salaryFee = salaryFee;
}
```

❖ The owner can change the transaction limit minimum up to 0.01%



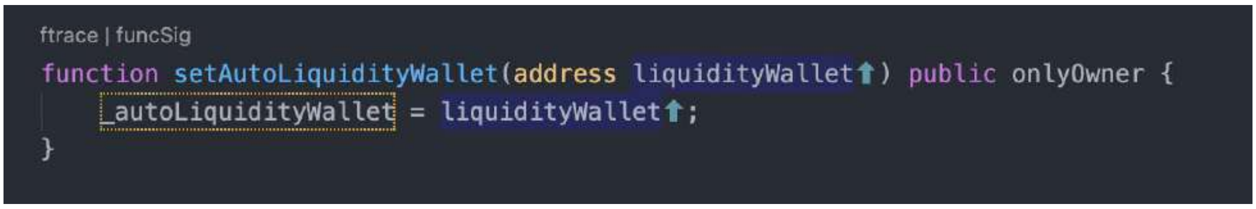
```
ftrace | funcSig
function setTransactionLimit(uint256 limit) public onlyOwner {
    require(
        limit >= 1 && limit <= 10000,
        "Limit must be greater than 0.01%"
    );
    _transactionLimit = _totalTokens / limit;
}
```

❖ The owner can change swap point



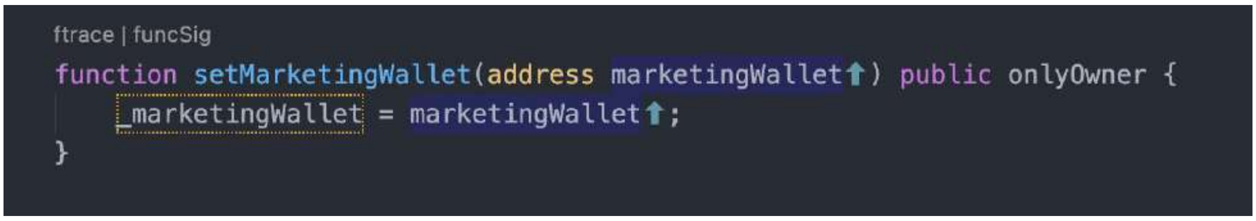
```
ftrace | funcSig
function setTokenSwapThreshold(uint256 threshold) public onlyOwner {
    require(threshold > 0, "Threshold must be greater than 0");
    _tokenSwapThreshold = threshold;
}
```

❖ The owner can change the liquidity receive wallet



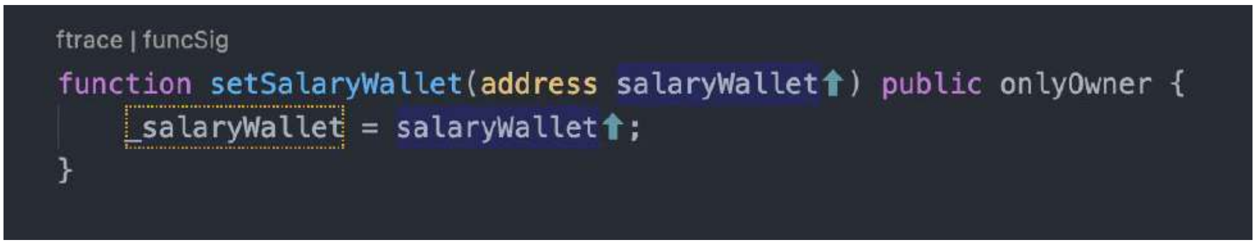
```
ftrace | funcSig
function setAutoLiquidityWallet(address liquidityWallet) public onlyOwner {
    _autoLiquidityWallet = liquidityWallet;
}
```

❖ The owner can change the marketing wallet address



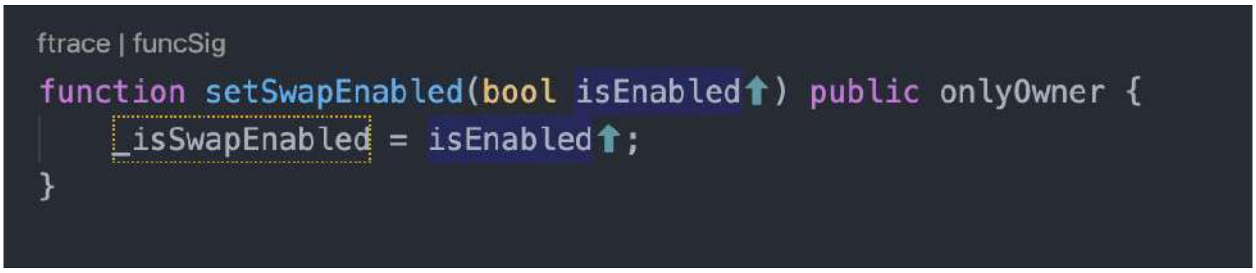
```
ftrace | funcSig
function setMarketingWallet(address marketingWallet) public onlyOwner {
    _marketingWallet = marketingWallet;
}
```

❖ The owner can change the salary wallet address



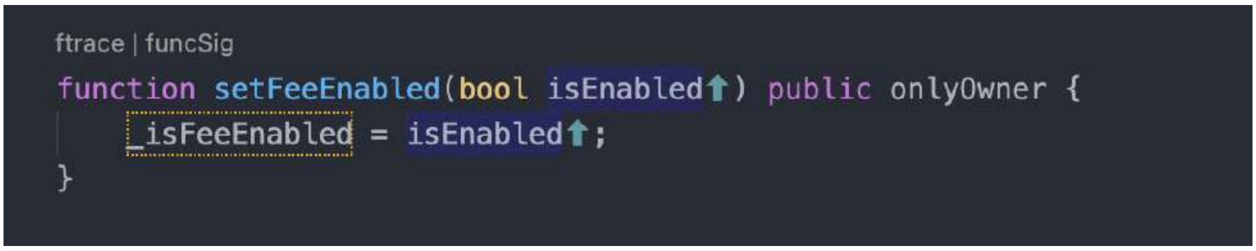
```
ftrace | funcSig
function setSalaryWallet(address salaryWallet) public onlyOwner {
    _salaryWallet = salaryWallet;
}
```

❖ The owner can enable/disable swapping



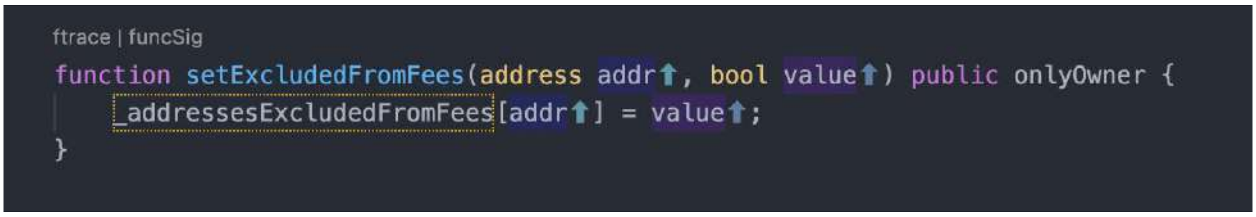
```
ftrace | funcSig
function setSwapEnabled(bool isEnabled) public onlyOwner {
    _isSwapEnabled = isEnabled;
}
```

❖ The owner can enable/disable fees



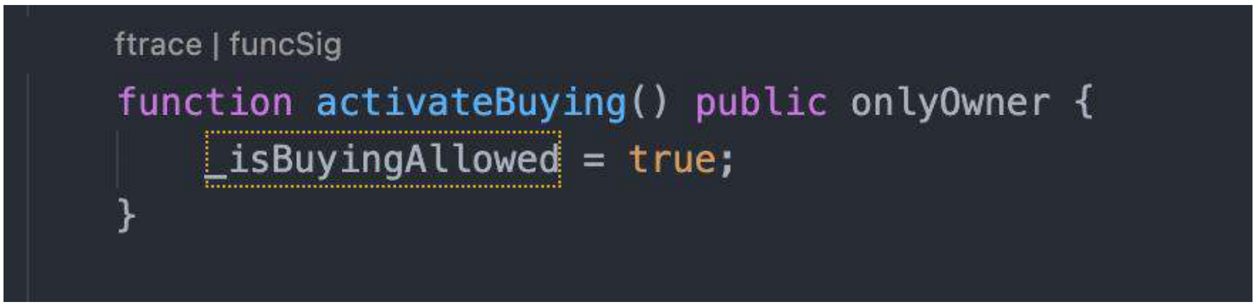
```
ftrace | funcSig
function setFeeEnabled(bool isEnabled) public onlyOwner {
    _isFeeEnabled = isEnabled;
}
```

❖ The owner can include/exclude wallets from fees



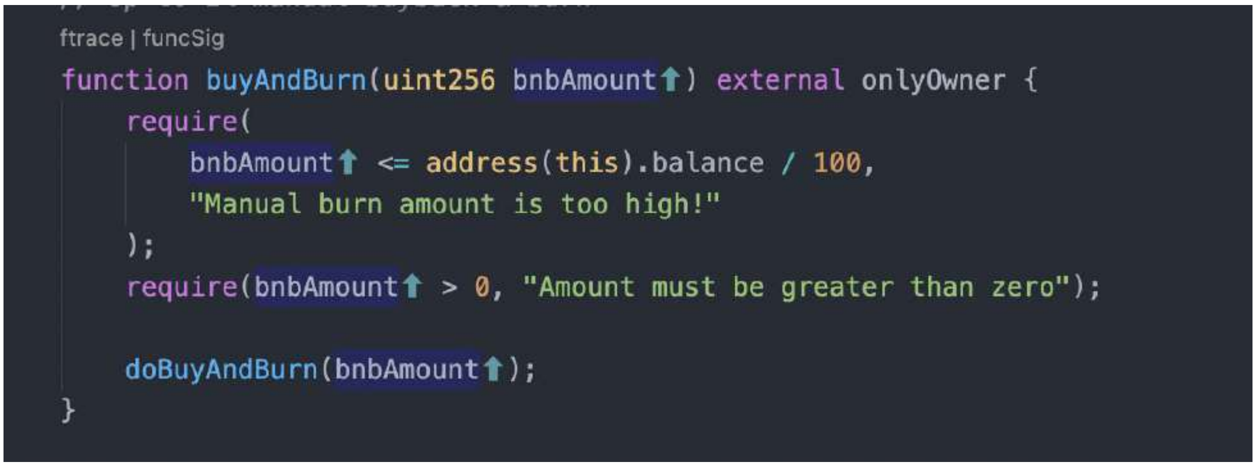
```
ftrace | funcSig
function setExcludedFromFees(address addr, bool value) public onlyOwner {
    _addressesExcludedFromFees[addr] = value;
}
```

❖ The owner can activate buying



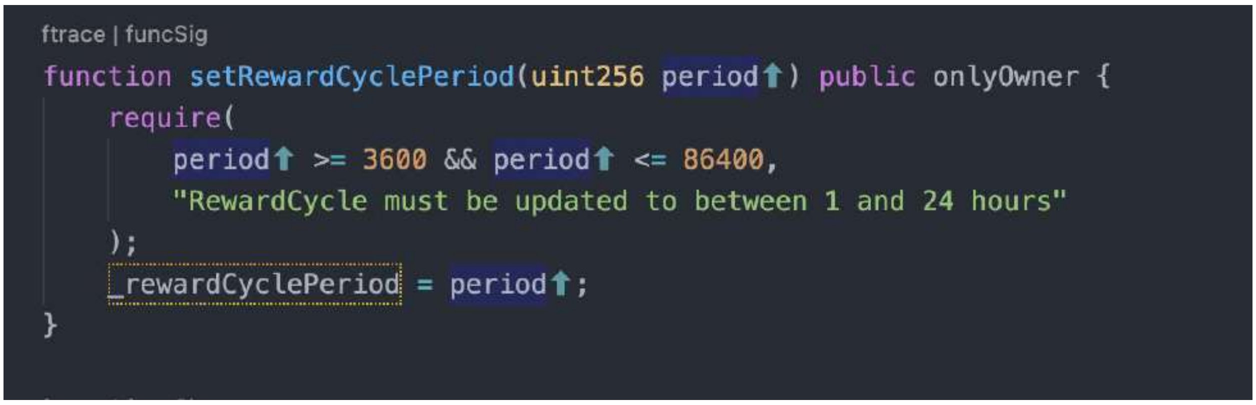
```
ftrace | funcSig
function activateBuying() public onlyOwner {
    _isBuyingAllowed = true;
}
```

❖ The owner can manually buy back and burn tokens from contract BNB balance maximum up to 1% bnb



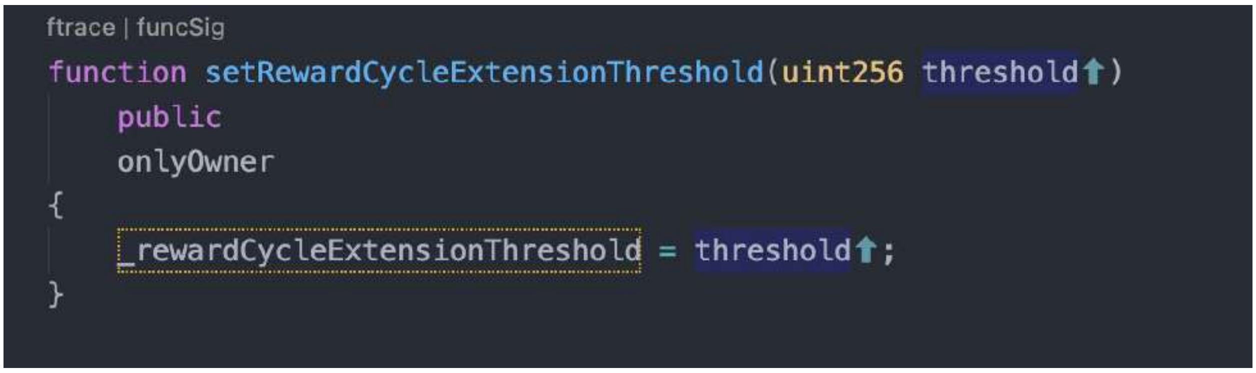
```
ftrace | funcSig
function buyAndBurn(uint256 bnbAmount) external onlyOwner {
    require(
        bnbAmount <= address(this).balance / 100,
        "Manual burn amount is too high!"
    );
    require(bnbAmount > 0, "Amount must be greater than zero");

    doBuyAndBurn(bnbAmount);
}
```

❖ The owner can change reward time period from 1 hour to 24 hours



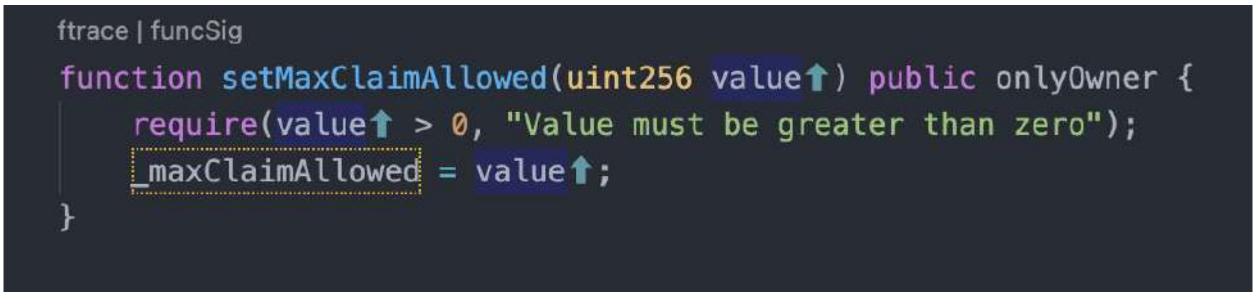
```
ftrace | funcSig
function setRewardCyclePeriod(uint256 period) public onlyOwner {
    require(
        period >= 3600 && period <= 86400,
        "RewardCycle must be updated to between 1 and 24 hours"
    );
    _rewardCyclePeriod = period;
}
```

❖ The owner can change reward cycle extension threshold (If user transfers more than threshold percentage of token, their reward claim time will be extended as a percentage of users' balance and transfer amount)



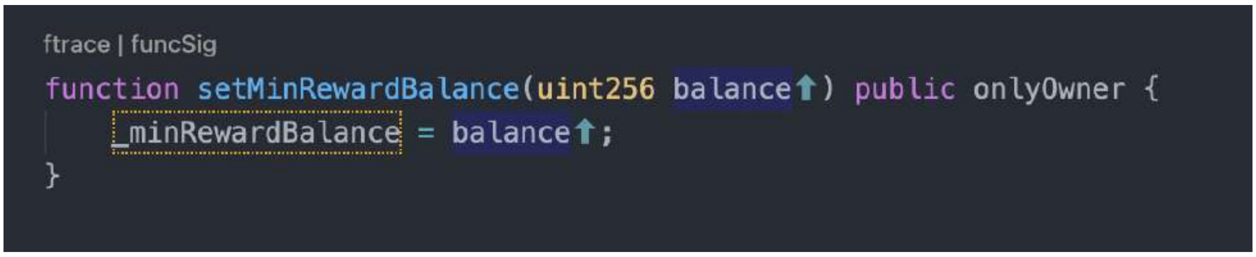
```
ftrace | funcSig
function setRewardCycleExtensionThreshold(uint256 threshold)
    public
    onlyOwner
{
    _rewardCycleExtensionThreshold = threshold;
}
```

❖ The owner can change max claim BNB amount at a time



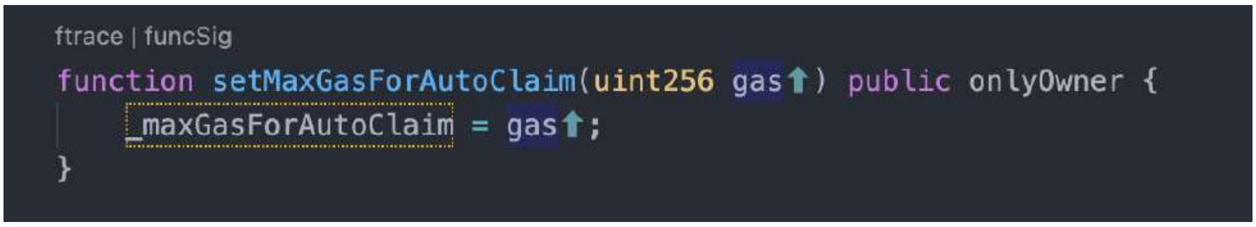
```
ftrace | funcSig
function setMaxClaimAllowed(uint256 value) public onlyOwner {
    require(value > 0, "Value must be greater than zero");
    _maxClaimAllowed = value;
}
```

❖ The owner can change minimum token amount to hold for eligible for the rewards



```
ftrace | funcSig
function setMinRewardBalance(uint256 balance) public onlyOwner {
    _minRewardBalance = balance;
}
```

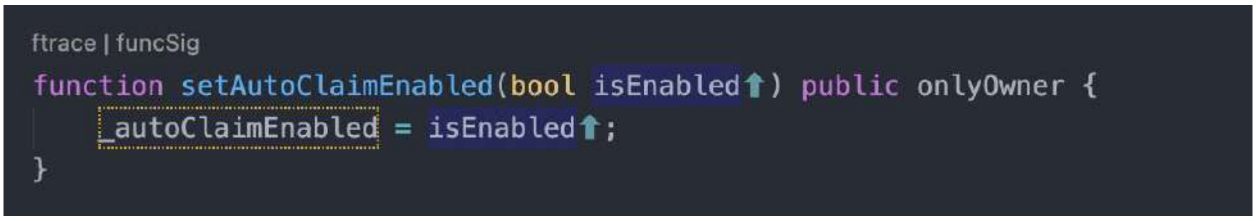
❖ The owner can change max gas limit for auto claim



ftrace | funcSig

```
function setMaxGasForAutoClaim(uint256 gas) public onlyOwner {
    _maxGasForAutoClaim = gas;
}
```

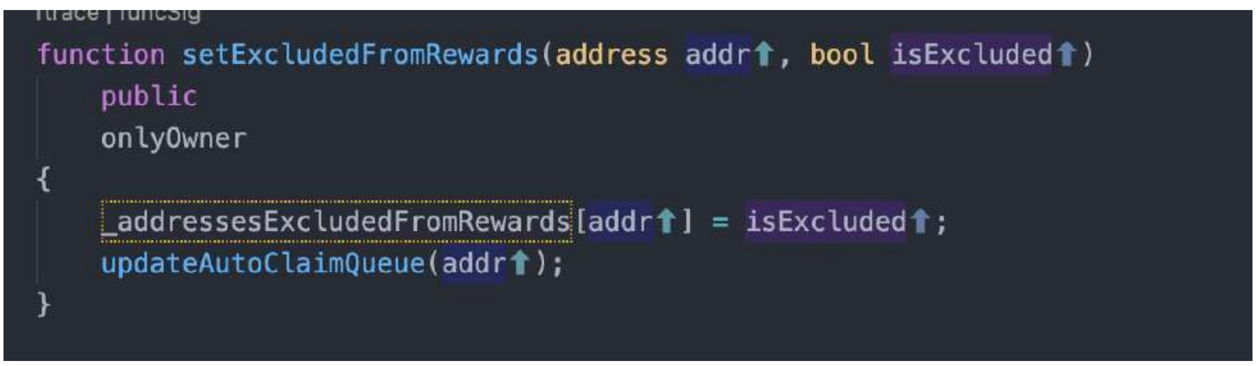
❖ The owner can enable/disable auto claim



ftrace | funcSig

```
function setAutoClaimEnabled(bool isEnabled) public onlyOwner {
    _autoClaimEnabled = isEnabled;
}
```

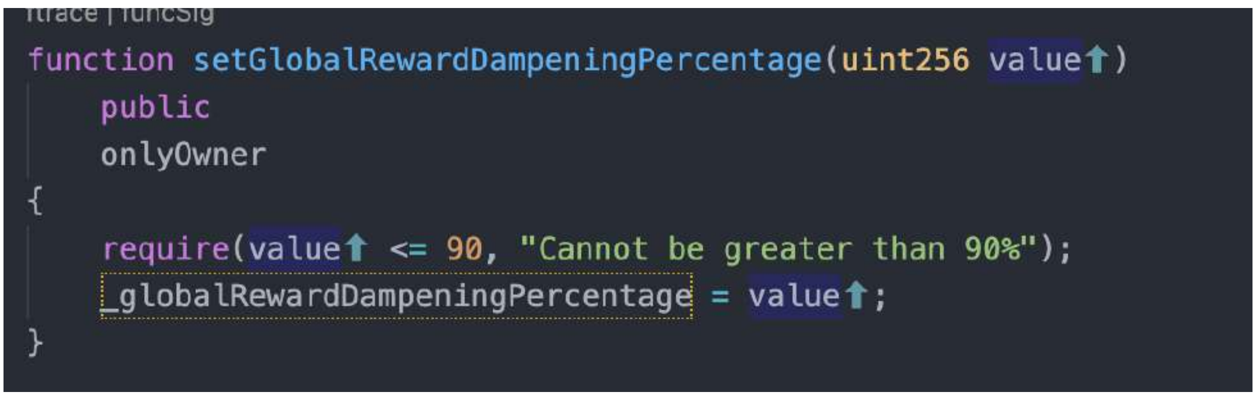
❖ The owner can include/exclude wallets from rewards



ftrace | funcSig

```
function setExcludedFromRewards(address addr, bool isExcluded)
    public
    onlyOwner
{
    _addressesExcludedFromRewards[addr] = isExcluded;
    updateAutoClaimQueue(addr);
}
```

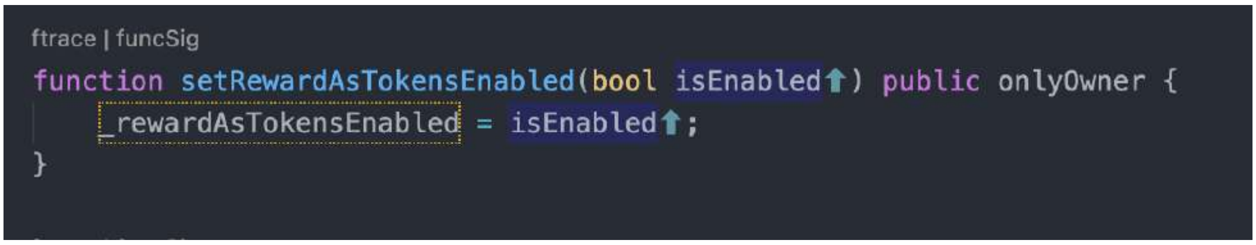
❖ The owner can change reward dumping percentage maximum up to 90% (If dumping percentage is 90, then only 10% of bnb will be rewarded)



ftrace | funcSig

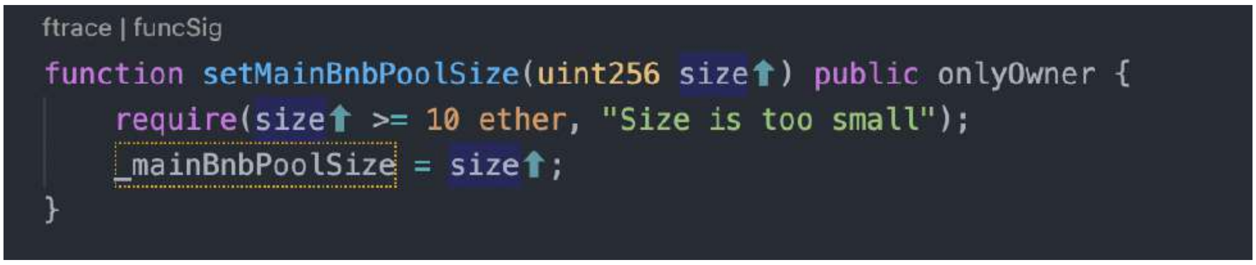
```
function setGlobalRewardDampeningPercentage(uint256 value)
    public
    onlyOwner
{
    require(value <= 90, "Cannot be greater than 90%");
    _globalRewardDampeningPercentage = value;
}
```

- ❖ The owner can enable/disable reward as token (Holders can get any number of percentages from their BNB reward value as tokens, if owner disable this, they will get full amount as BNB and if it's enabled, they will receive both BNBs and tokens according to their percentages)



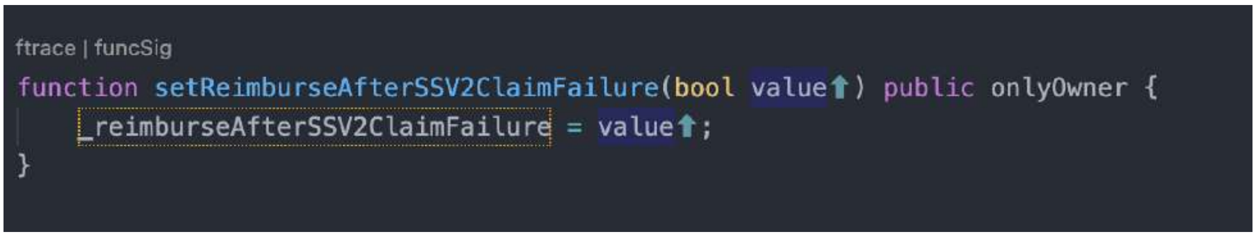
```
function setRewardAsTokensEnabled(bool isEnabled) public onlyOwner {
    _rewardAsTokensEnabled = isEnabled;
}
```

- ❖ The owner can change main pool size minimum up to 10 BNB (main pool size is maximum bnb amount allocate for rewards)



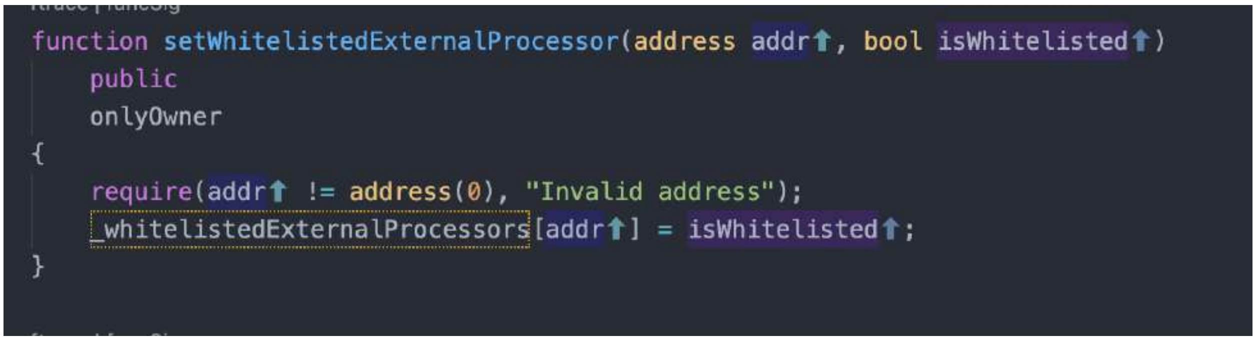
```
function setMainBnbPoolSize(uint256 size) public onlyOwner {
    require(size >= 10 ether, "Size is too small");
    _mainBnbPoolSize = size;
}
```

- ❖ The owner can enable/disable reimburse after fail (If this is enabled, if sending tokens fails, holders will receive their full reward in BNBs)



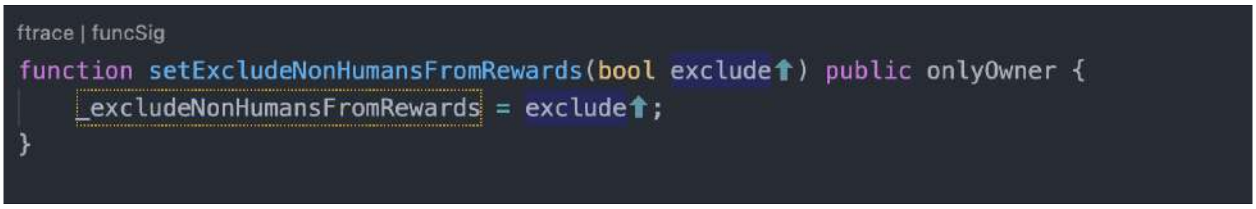
```
function setReimburseAfterSSV2ClaimFailure(bool value) public onlyOwner {
    _reimburseAfterSSV2ClaimFailure = value;
}
```

- ❖ The owner can whitelist wallets as external processor (If whitelisted wallet manually process, reward claim contract will return gas fee back)



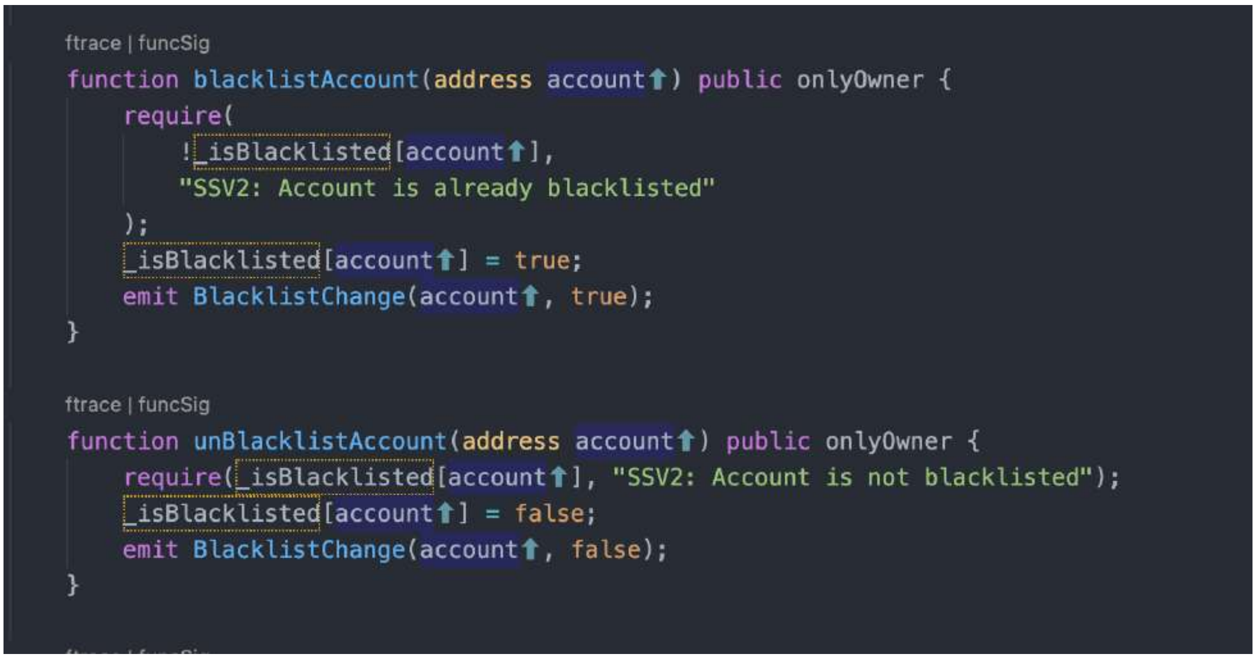
```
function setWhitelistedExternalProcessor(address addr, bool isWhitelisted)
    public
    onlyOwner
{
    require(addr != address(0), "Invalid address");
    _whitelistedExternalProcessors[addr] = isWhitelisted;
}
```

❖ The owner can enable/disable contracts from rewards



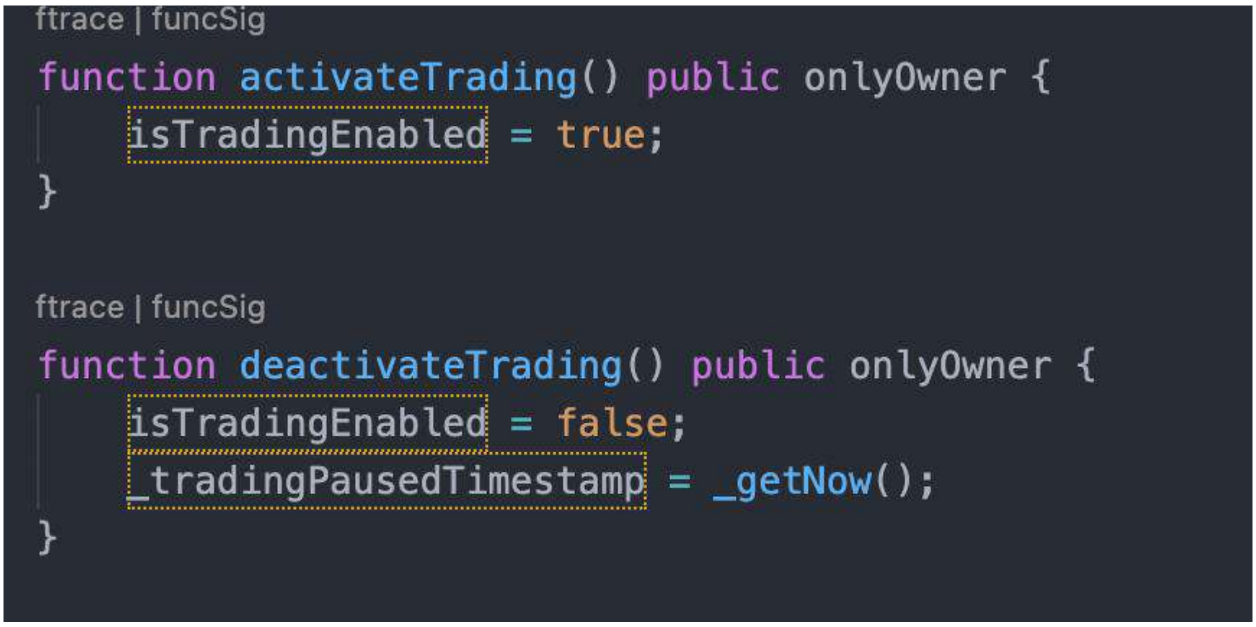
```
ftrace | funcSig
function setExcludeNonHumansFromRewards(bool exclude) public onlyOwner {
    _excludeNonHumansFromRewards = exclude;
}
```

❖ The owner can blacklist and unblack list wallets



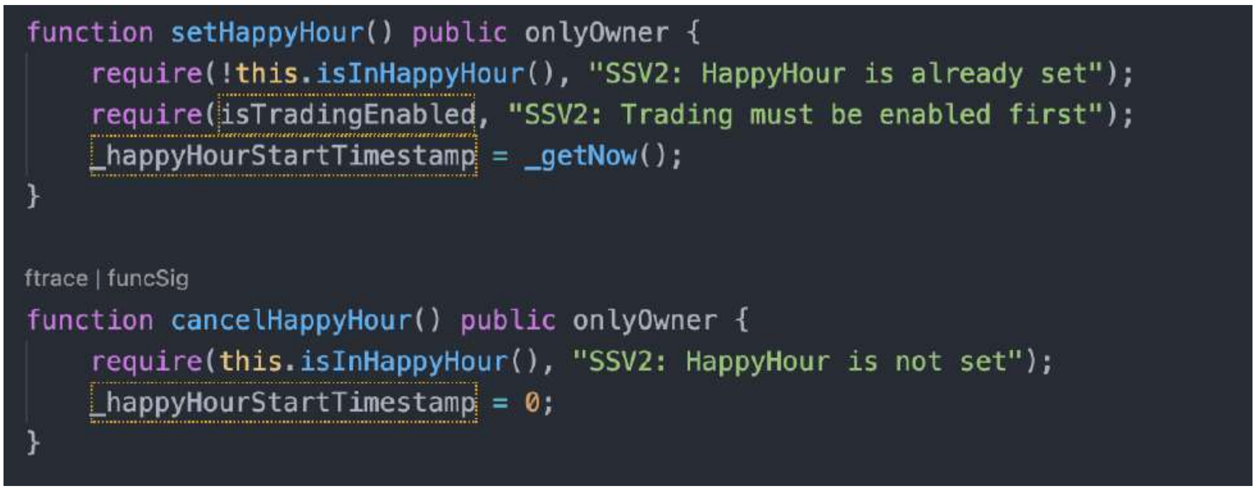
```
ftrace | funcSig
function blacklistAccount(address account) public onlyOwner {
    require(
        !_isBlacklisted[account],
        "SSV2: Account is already blacklisted"
    );
    _isBlacklisted[account] = true;
    emit BlacklistChange(account, true);
}

ftrace | funcSig
function unBlacklistAccount(address account) public onlyOwner {
    require(_isBlacklisted[account], "SSV2: Account is not blacklisted");
    _isBlacklisted[account] = false;
    emit BlacklistChange(account, false);
}
```

❖ The owner can activate and deactivate trading



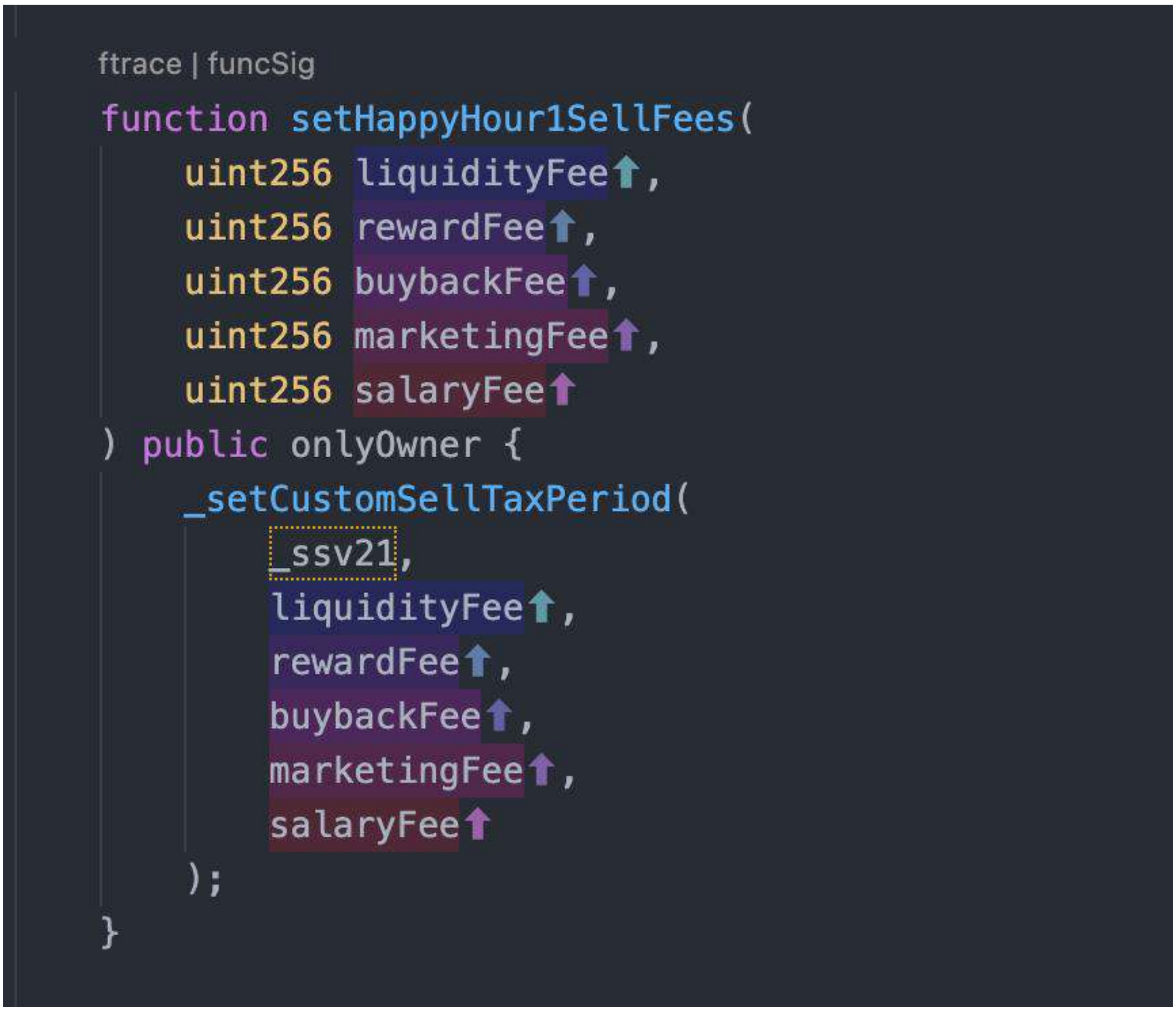
```
ftrace | funcSig
function activateTrading() public onlyOwner {
    isTradingEnabled = true;
}

ftrace | funcSig
function deactivateTrading() public onlyOwner {
    isTradingEnabled = false;
    _tradingPausedTimestamp = _getNow();
}
```

❖ The owner can set and cancel happy hour



```
function setHappyHour() public onlyOwner {
    require(!this.isInHappyHour(), "SSV2: HappyHour is already set");
    require(isTradingEnabled, "SSV2: Trading must be enabled first");
    _happyHourStartTimestamp = _getNow();
}

ftrace | funcSig
function cancelHappyHour() public onlyOwner {
    require(this.isInHappyHour(), "SSV2: HappyHour is not set");
    _happyHourStartTimestamp = 0;
}
```

❖ The owner can change all buy and sell fees to happy hour 1,2 and 3



```
ftrace | funcSig
function setHappyHour1SellFees(
    uint256 liquidityFee,
    uint256 rewardFee,
    uint256 buybackFee,
    uint256 marketingFee,
    uint256 salaryFee
) public onlyOwner {
    _setCustomSellTaxPeriod(
        _ssv21,
        liquidityFee,
        rewardFee,
        buybackFee,
        marketingFee,
        salaryFee
    );
}
```

# 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: **2**

Solidity code functional issue level: **PASSED**

Number of owner privileges: **31**

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

Smart contract active ownership: **YES**