RugFreeCoins Audit

Dark Dao Token

Smart Contract Security Audit

April 03, 2022

Contents

Audit details

Audited project
Dark Dao Token

Contract Address

0x49B1C4387bA08976513F81b27d05265F6D9267fa

Client contact

Dark Dao Team

Blockchain

Binance smart chain

Project website

https://dark-dao.net/

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 Dark Dao Team to perform an audit of the smart contract.

https://bscscan.com/token/0x49B1C4387bA08976513F81b27d05265F6D9267fa

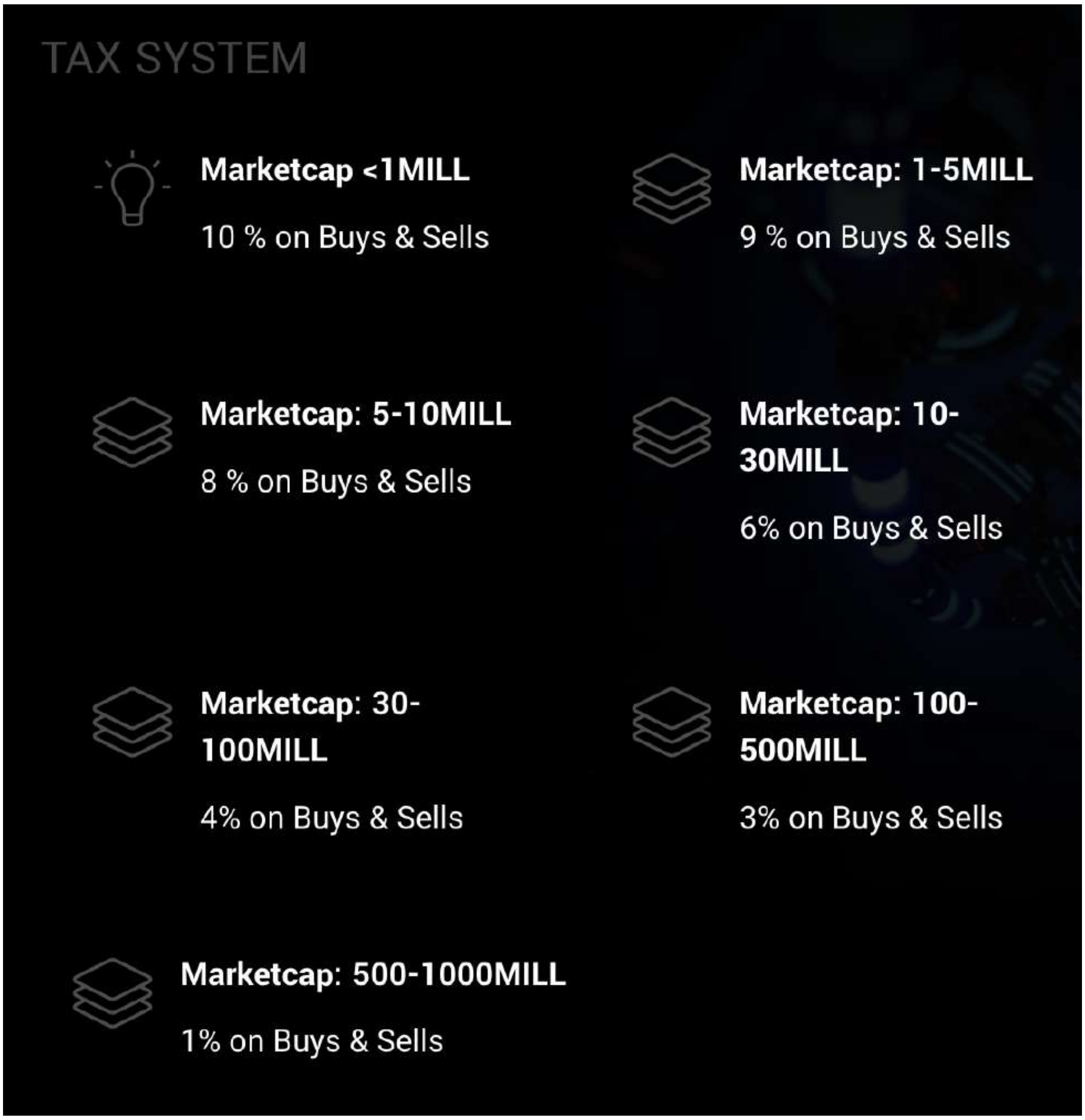
The focus of this audit is to verify that the smart contract is secure, resilient, and working according to the specifications.

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

Dark Dao is a token built on the Binance Smart Chain that is with an innovative investment use case the main purpose of which is to seek out constant revenue sources, which in turn powers P2E high-end gaming experience, which creates addiction from the first second of playing. They are building a game that sets new levels of gaming fun combined with cryptocurrencies, as well as NFT and land staking. Each transaction, purchase, and sale incur a 10% fee.



TAX SYSTEM

Marketcap <1MILL

10 % on Buys & Sells

Marketcap: 1-5MILL

9 % on Buys & Sells

Marketcap: 5-10MILL

8 % on Buys & Sells

Marketcap: 10-30MILL

6% on Buys & Sells

Marketcap: 30-100MILL

4% on Buys & Sells

Marketcap: 100-500MILL

3% on Buys & Sells

Marketcap: 500-1000MILL

1% on Buys & Sells

Features

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

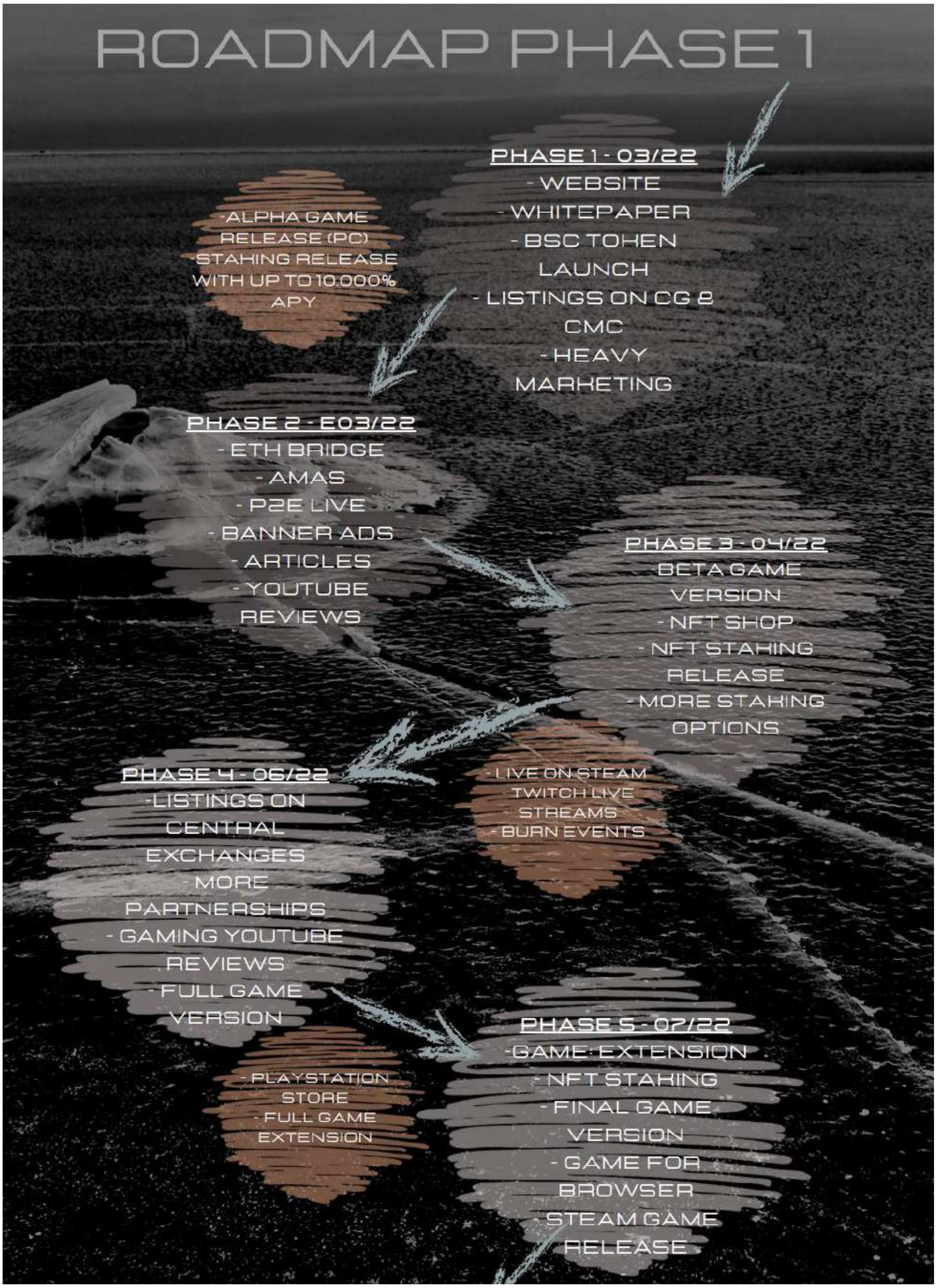
- The **sustainability fee of 3% when buying and selling for marketing, and 1% when buying and selling for dev** is what allows Dark Dao to hold the aforementioned promise. Tokens will be swapped into BUSD and will be sent to a marketing wallet and dev wallet. This way, Dark Dao will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

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

- 1% when buying and selling are converted to BUSD and allocated for team funds.

- 2% when buying and selling is converted to BUSD and sent to the game wallet for P2E game rewards and development.

Roadmap



ROADMAP PHASE 1

PHASE 1 - 03/22
- WEBSITE
- WHITEPAPER
- BSC TOKEN LAUNCH
- LISTINGS ON CG & CMC
- HEAVY MARKETING

-ALPHA GAME RELEASE (PC)
-STAKING RELEASE WITH UP TO 10.000% APY

PHASE 2 - E03/22
- ETH BRIDGE
- AMAS
- P2E LIVE
- BANNER ADS
- ARTICLES
- YOUTUBE REVIEWS

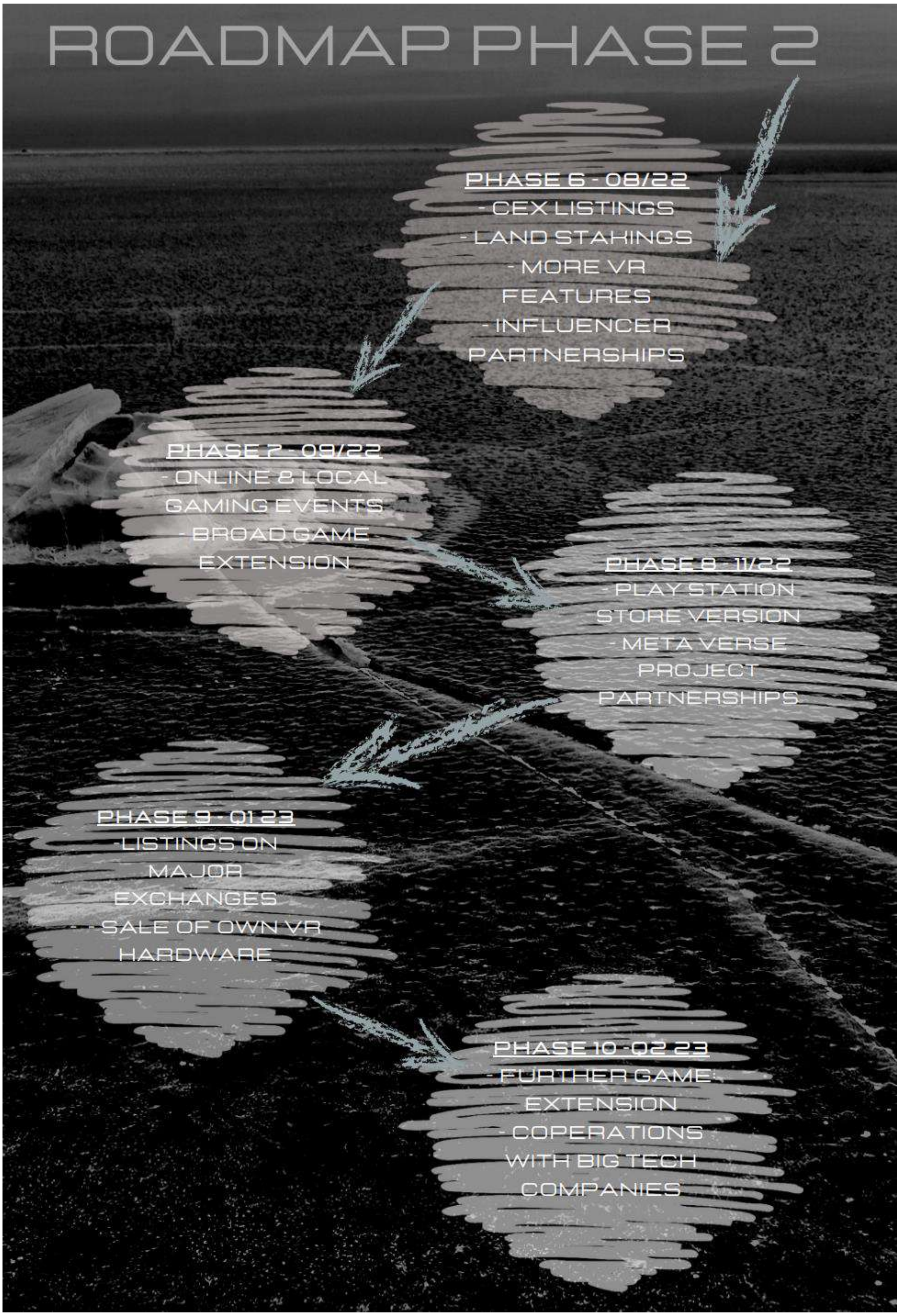
PHASE 3 - 04/22
-BETA GAME VERSION
- NFT SHOP
- NFT STAKING RELEASE
- MORE STAKING OPTIONS

PHASE 4 - 06/22
-LISTINGS ON CENTRAL EXCHANGES
- MORE PARTNERSHIPS
- GAMING YOUTUBE REVIEWS
- FULL GAME VERSION

- LIVE ON STEAM
TWITCH LIVE STREAMS
- BURN EVENTS

- PLAYSTATION STORE
- FULL GAME EXTENSION

PHASE 5 - 07/22
-GAME EXTENSION
- NFT STAKING
- FINAL GAME VERSION
- GAME FOR BROWSER
- STEAM GAME RELEASE



ROADMAP PHASE 2

PHASE 6 - 08/22
- CEX LISTINGS
- LAND STAKINGS
- MORE VR FEATURES
- INFLUENCER PARTNERSHIPS

PHASE 7 - 09/22
- ONLINE & LOCAL GAMING EVENTS
- BROAD GAME EXTENSION

PHASE 8 - 11/22
- PLAY STATION STORE VERSION
- META VERSE PROJECT PARTNERSHIPS

PHASE 9 - Q1 23
-LISTINGS ON MAJOR EXCHANGES
- -SALE OF OWN VR HARDWARE

PHASE 10 - Q2 23
- FURTHER GAME EXTENSION
- COPERATIONS WITH BIG TECH COMPANIES

Tokenomics

11% fee when buying and selling

- 2% of trade goes to holders’ pockets in BUSD.
- 3% of trade goes to the marketing wallet
- 1% of trade goes to the Dev wallet
- 2% of trade goes to the Team wallet
- 2% of trade goes to the Game wallet
- 2% of trade goes to the liquidity pool.

Target market and the concept

Target market

- Anyone who's interested in the Crypto space with long-term investment plans.
- Anyone who's ready to earn a passive income in BUSD by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who is ready to stake tokens and win rewards.
- Anyone who is ready to be part of the P2E game and win rewards.
- Anyone who's interested in collecting NFTs or trading NFTs.
- Anyone who's interested in taking part in the future plans of the Dark Dao token.
- Anyone who's interested in making financial transactions with any other party using BUSD or Dark Dao as the currency.

Core concept

The Dark Dao reward system

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

Sustainable mechanism

The **sustainability fee of 3% when buying and selling for marketing and 1% when buying and selling for dev** is what allows Dark Dao to promote the token and use funds to further the development of the platform. Tokens will be swapped into BUSD and will be sent to marketing and development wallets. This way, Dark Dao will have access to the funds without selling tokens as the traditional way, which will enable them to consume funds without hurting the project.

1% when buying and selling are converted to BUSD and allocated for team funds to pay salaries for the team members.

2% when buying and selling is converted to BUSD and sent to the game wallet for P2E game rewards and development.

The liquidity fee of 2%, is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

VISION

WE ARE DEVELOPING AI & VR EMBEDDED GAME PILLARS TO GIVE USERS A NEXT LEVEL GAMING EXPERIENCE.

OUR VISION IS TO DEVELOP A VR GAMING EXPERIENCE WHERE YOU MERGE WITH YOUR IN GAME CHARACTER BY TRAINING AND ADOPTING THEIR MOVEMENTS AS YOUR OWN.

FEATURES

GENERAL

2% REFLECTIONS FOR PASSIVE INCOME

CONTRACT CHECKER AS DEVELOPER & PINKSALE PARTNER

INITIAL LP LOCK FOR 12 MONTHS (EXTENSION TO 5 YEARS WITH FULL GAME RELEASE)

BSC CONTRACT WITH ETH & AVAX BRIDGE

UTILITIES

PLAY TO EARN VR GAMING

ROBOTER & LAND NFTS WITH IN-GAME USAGE

STAKING WITH UP TO 10.000% APY AND VESTING

SOCIAL DAO CLANS & IN-GAME COMMUNITIES

MARKETING

MASSIVE ADVERTISMENTS

LISTINGS ON CG, CMC & OTHER EXCHANGES

CELEBRITY PROMOTIONS

SMART CONTRACT AUDIT

TRENDING PROMOTIONS

COMMUNITY GIVEAWAYS

Potential to grow with score points

1.	Project efficiency	10/10
2.	Project uniqueness	9/10
3	Information quality	10/10
4	Service quality	9/10
5	System quality	10/10
6	Impact on the community	9/10
7	Impact on the business	10/10
8	Preparing for the future	10/10
Total Points		**9.625/10**

Contract details

Token contract details for 03rd April 2022

Contract name	Dark Dao
Contract address	0x49B1C4387bA08976513F81b27d05265F6D9267fa
Token supply	50,000,000
Token ticker	$DDAO
Decimals	9
Token holders	1
Transaction count	2
Dividend tracker	0xaf9b0b2d4482ea7e9c1bb07e0e10d945c66d45b0
Dev wallet 1	0x349f69e52e0b34f6b3ddbc7a0f8418477e8087a0
Dev wallet 2	0x77df19669310ad06959c3284ff5c27d9fe2ca1b7
Game fee receiver	0x686271159b9258f994c725742b098ee721d5c3f0
Marketing wallet	0x26d09311646acb6d951e7d73fe201f7168d98aa6
Contract deployer address	0xD9EA912E0169388dfAe2fADfEfaaca85dC505066
Contract’s current owner address	0xd9ea912e0169388dfae2fadfefaaca85dc505066

Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	pass
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	pass
		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
IUniswapV2 Factory	**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❗
IUniswapV2 Router01	**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❗
IUniswapV2Router02	**Interface**	**IUniswapV2Router01**		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗

└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
SafeMath	**Library**			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
IERC20	**Interface**			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗

└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IDividendDistributor	Interface			
└	setDistributionCriteria	External ❗	🛑	NO❗
└	setShare	External ❗	🛑	NO❗
└	deposit	External ❗	🛑	NO❗
└	process	External ❗	🛑	NO❗
└	purge	External ❗	🛑	NO❗
DividendDistributor	**Implementation**	**IDividendDistributor**		
└		Public ❗	🛑	NO❗
└		External ❗	💵	NO❗
└	setDistributionCriteria	External ❗	🛑	onlyToken
└	purge	External ❗	🛑	onlyToken
└	setShare	External ❗	🛑	onlyToken
└	deposit	External ❗	🛑	onlyToken
└	process	External ❗	🛑	onlyToken
└	shouldDistribute	Internal 🔒		
└	distributeDividend	Internal 🔒	🛑	

└	claimDividend	External ❗	🛑	NO❗
└	getUnpaidEarnings	Public ❗		NO❗
└	getHolderDetails	Public ❗		NO❗
└	getCumulativeDividends	Internal 🔒		
└	getLastProcessedIndex	External ❗		NO❗
└	getNumberOfTokenHolders	External ❗		NO❗
└	getShareHoldersList	External ❗		NO❗
└	totalDistributedRewards	External ❗		NO❗
└	addShareholder	Internal 🔒	🛑	
└	removeShareholder	Internal 🔒	🛑	
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	

DDAO	**Implementation**	**IERC20, Ownable**		
└		Public ❗️	🛑	NO❗️
└		External ❗️	💵	NO❗️
└	totalSupply	External ❗️		NO❗️
└	name	Public ❗️		NO❗️
└	symbol	Public ❗️		NO❗️
└	decimals	Public ❗️		NO❗️
└	balanceOf	Public ❗️		NO❗️
└	getHolderDetails	Public ❗️		NO❗️
└	getLastProcessedIndex	Public ❗️		NO❗️
└	getNumberOfTokenHolders	Public ❗️		NO❗️
└	totalDistributedRewards	Public ❗️		NO❗️
└	allowance	External ❗️		NO❗️
└	approve	Public ❗️	🛑	NO❗️
└	_approve	Internal 🔒	🛑	
└	approveMax	External ❗️	🛑	NO❗️
└	transfer	External ❗️	🛑	NO❗️
└	transferFrom	External ❗️	🛑	NO❗️
└	_transferFrom	Internal 🔒	🛑	
└	_basicTransfer	Internal 🔒	🛑	
└	shouldTakeFee	Internal 🔒		

└	takeFee	Internal 🔒	🛑	
└	shouldSwapBack	Internal 🔒		
└	clearStuckBalance	External ❗	🛑	onlyOwner
└	getBep20Tokens	External ❗	🛑	onlyOwner
└	updateBuyFees	Public ❗	🛑	onlyOwner
└	updateSellFees	Public ❗	🛑	onlyOwner
└	updateSwapPercentages	Public ❗	🛑	onlyOwner
└	enableTrading	Public ❗	🛑	onlyOwner
└	whitelistPreSale	Public ❗	🛑	onlyOwner
└	___claimRewards	Public ❗	🛑	NO❗
└	claimProcess	Public ❗	🛑	NO❗
└	blackListWallets	Public ❗	🛑	onlyOwner
└	isBlacklisted	Public ❗		NO❗
└	isRewardExclude	Public ❗		NO❗
└	isFeeExclude	Public ❗		NO❗
└	isMaxWalletExclude	Public ❗		NO❗
└	isMaxTxExcluded	Public ❗		NO❗
└	setIsMaxTxExempt	External ❗	🛑	onlyOwner
└	setMaxTxAmount	External ❗	🛑	onlyOwner
└	isExemptTimeLock	Public ❗		NO❗
└	changeSellCoolDownTime	Public ❗	🛑	onlyOwner

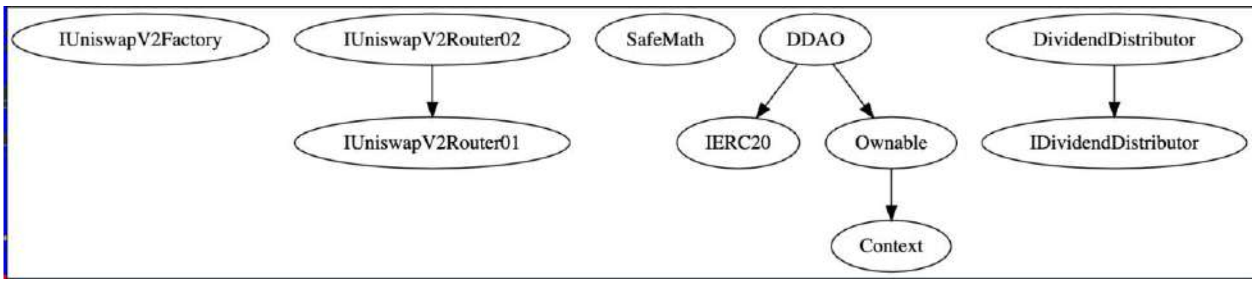
└	enableSellCollDown	Public ❗	🛑	onlyOwner
└	exemptTimeLock	Public ❗	🛑	onlyOwner
└	swapBackInBnb	Internal 🔒	🛑	swapping
└	swapAndLiquify	Private 🔐	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	swapTokensForTokens	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	setIsDividendExempt	External ❗	🛑	onlyOwner
└	setIsFeeExempt	External ❗	🛑	onlyOwner
└	setIsMaxWalletExempt	External ❗	🛑	onlyOwner
└	addAuthorizedWallets	External ❗	🛑	onlyOwner
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setGameWallet	External ❗	🛑	onlyOwner
└	setTeamWallet	External ❗	🛑	onlyOwner
└	setDevWallets	External ❗	🛑	onlyOwner
└	setMaxWalletToken	External ❗	🛑	onlyOwner
└	setSwapBackSettings	External ❗	🛑	onlyOwner
└	setDistributionCriteria	External ❗	🛑	onlyOwner
└	setDistributorSettings	External ❗	🛑	onlyOwner
└	purgeBeforeSwitch	Public ❗	🛑	onlyOwner
└	includeMeinRewards	Public ❗	🛑	NO❗

L	switchToken	Public ❗	🛑	onlyOwner

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

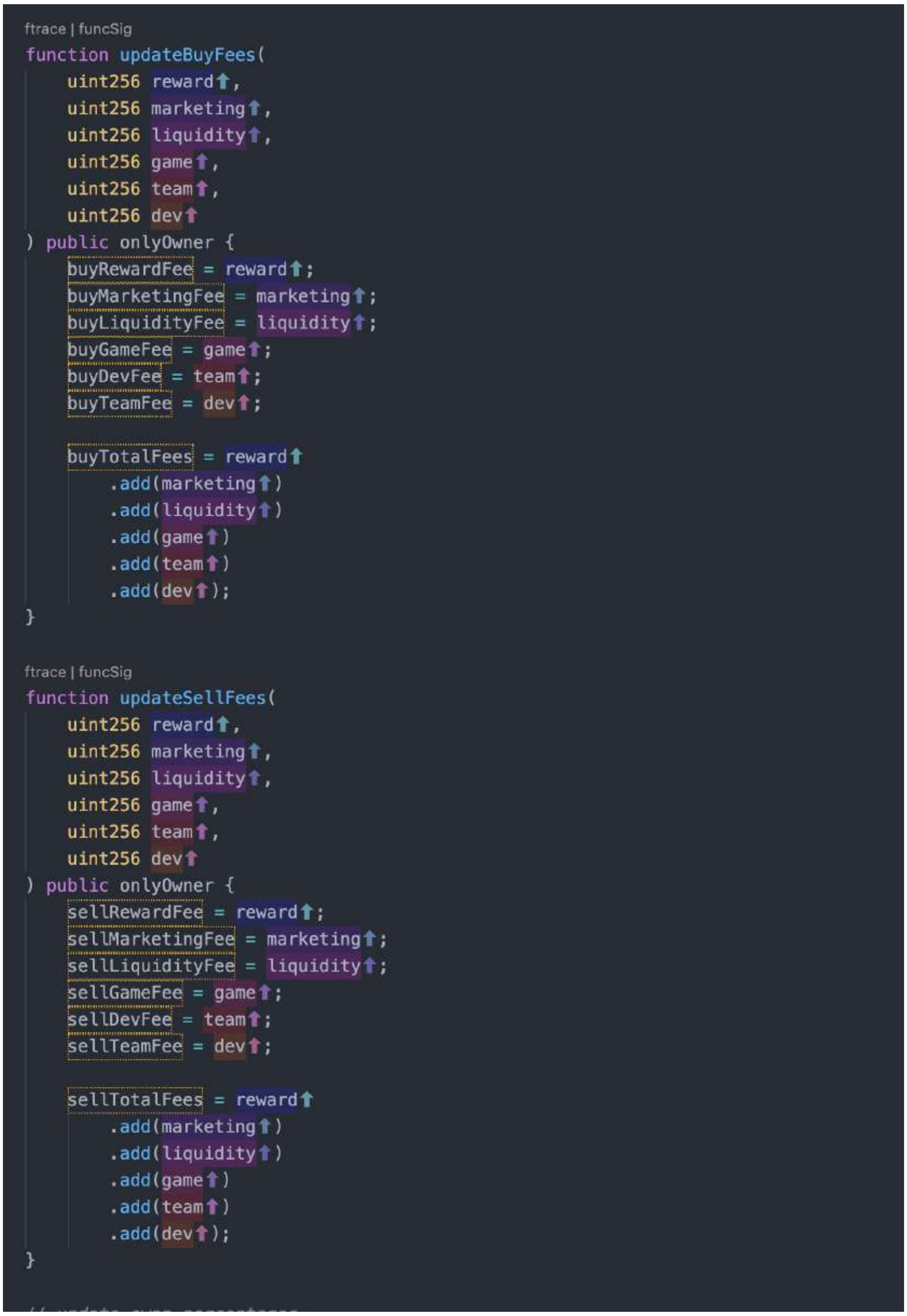
- **Low severity issues**

No medium severity issues found

- **Centralization risk**

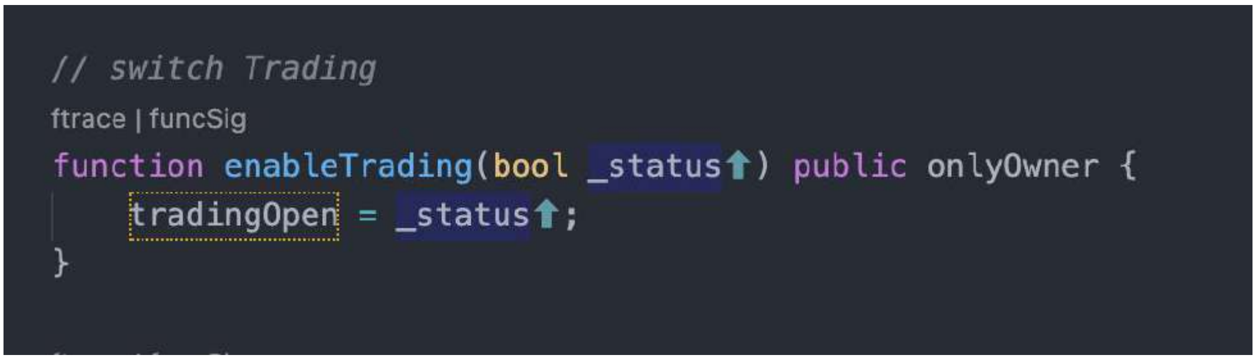
High issues

❖ The owner can change all fees without any limitation (can set 100%)



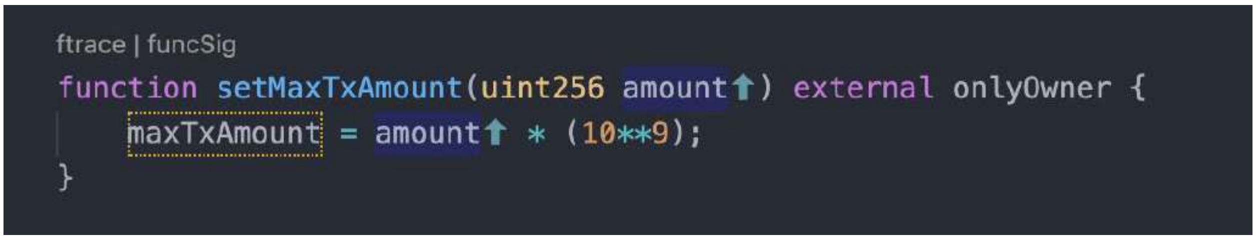
```
ftrace | funcSig
function updateBuyFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 game,
    uint256 team,
    uint256 dev
) public onlyOwner {
    buyRewardFee = reward;
    buyMarketingFee = marketing;
    buyLiquidityFee = liquidity;
    buyGameFee = game;
    buyDevFee = team;
    buyTeamFee = dev;

    buyTotalFees = reward
        .add(marketing)
        .add(liquidity)
        .add(game)
        .add(team)
        .add(dev);
}

ftrace | funcSig
function updateSellFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 game,
    uint256 team,
    uint256 dev
) public onlyOwner {
    sellRewardFee = reward;
    sellMarketingFee = marketing;
    sellLiquidityFee = liquidity;
    sellGameFee = game;
    sellDevFee = team;
    sellTeamFee = dev;

    sellTotalFees = reward
        .add(marketing)
        .add(liquidity)
        .add(game)
        .add(team)
        .add(dev);
}
```

❖ The owner can enable/disable trading anytime



```
// switch Trading
ftrace | funcSig
function enableTrading(bool _status) public onlyOwner {
    tradingOpen = _status;
}
```

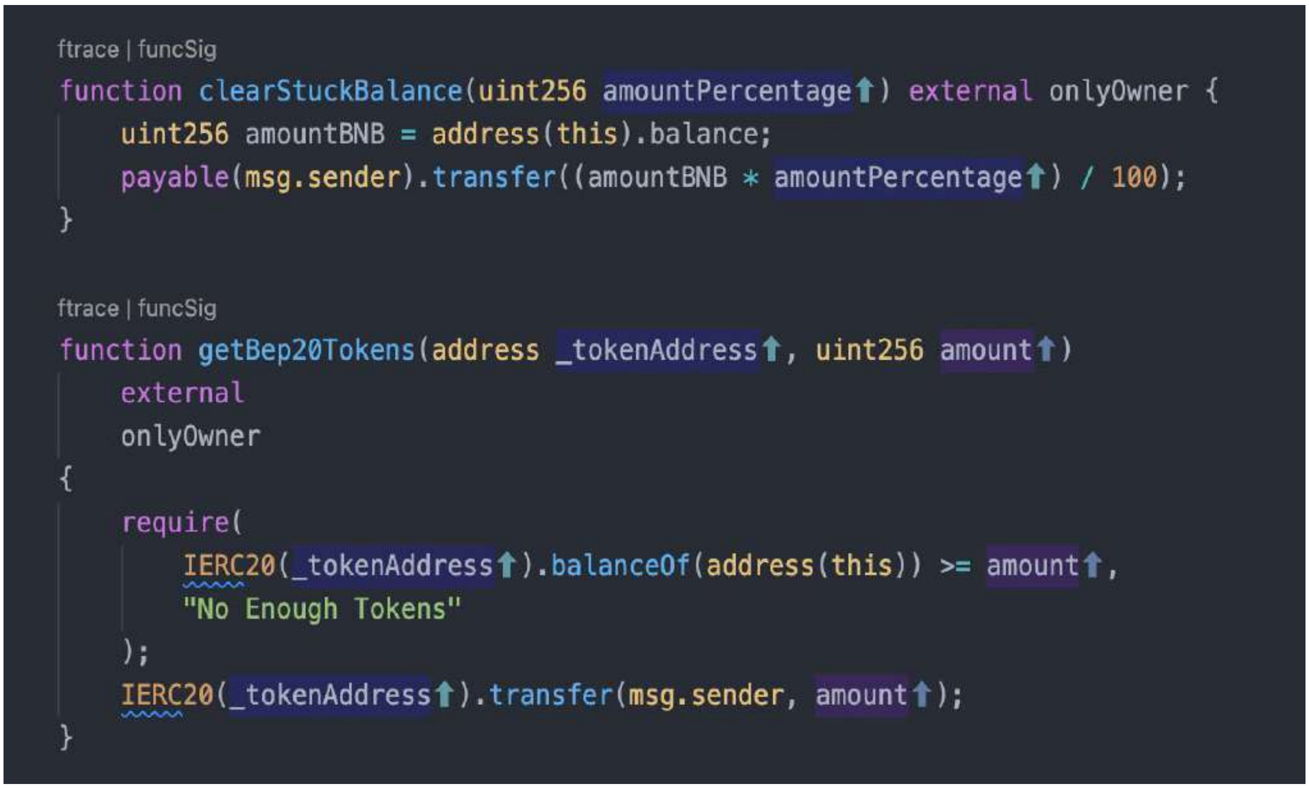
❖ The owner can change max transaction amount without minimum limit (can set 0)



```
ftrace | funcSig
function setMaxTxAmount(uint256 amount) external onlyOwner {
    maxTxAmount = amount * (10**9);
}
```

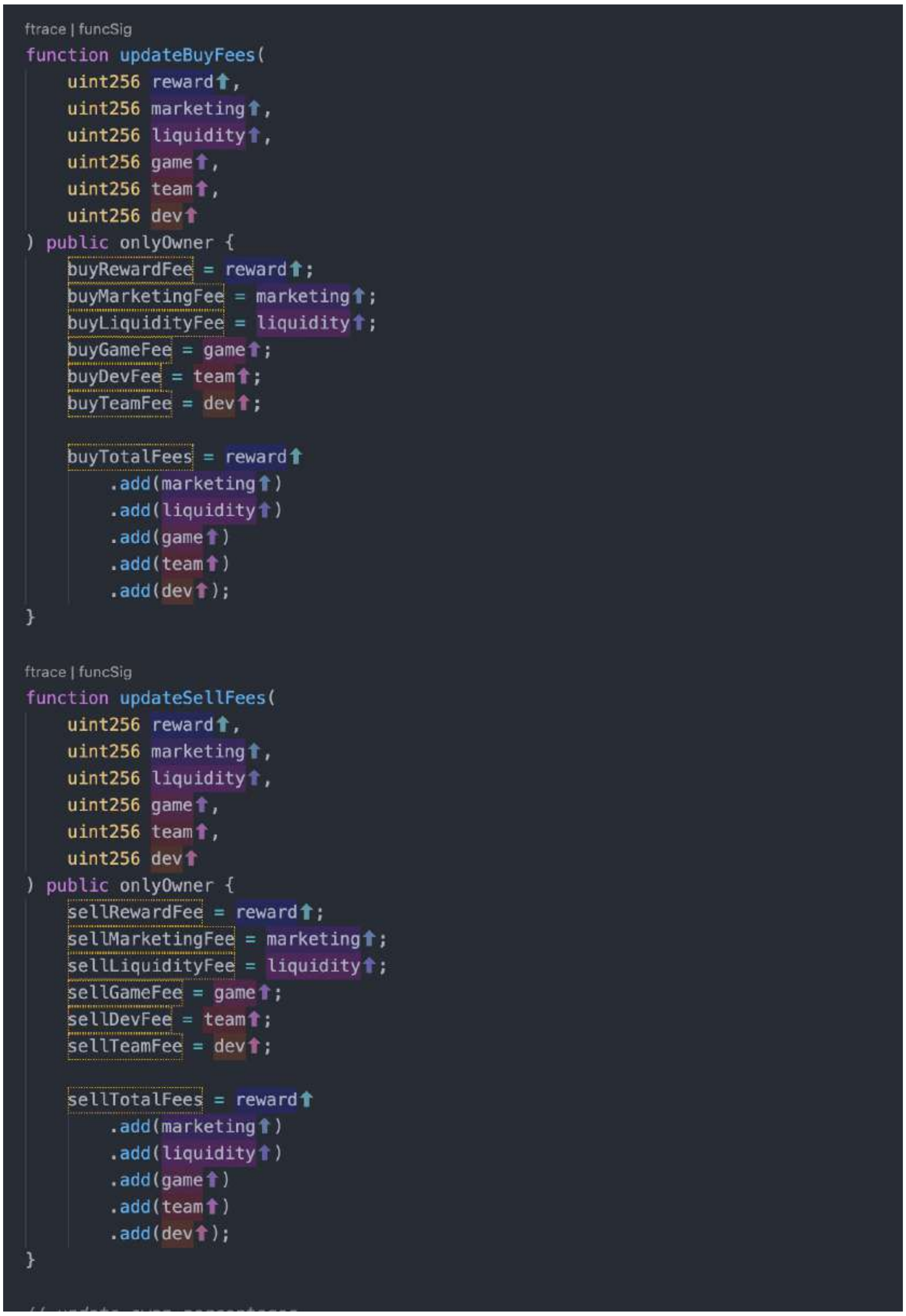
Owner privileges

- The owner can get BNB and bep20 token in contract to owner wallet



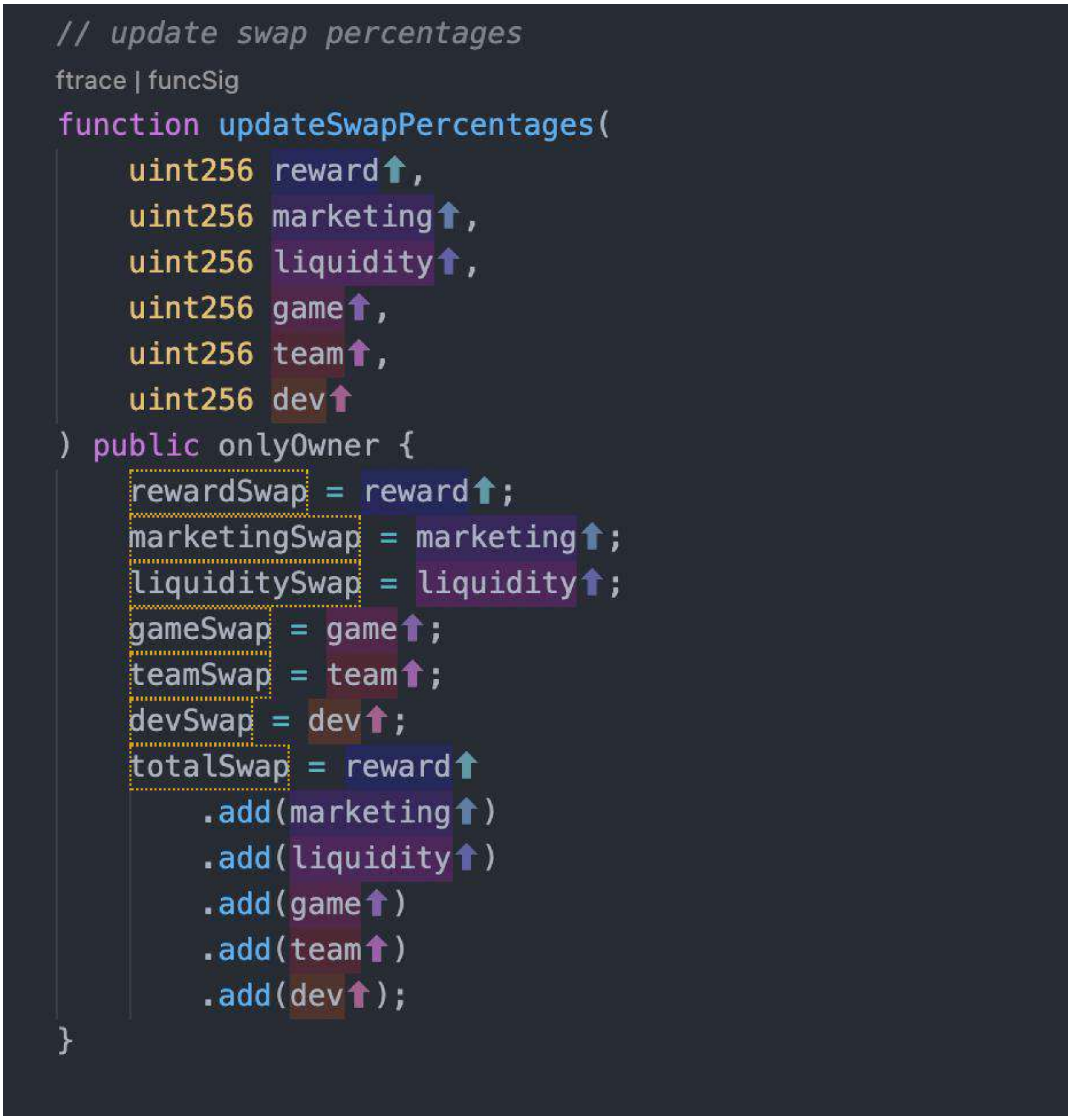
```
ftrace | funcSig
function clearStuckBalance(uint256 amountPercentage) external onlyOwner {
    uint256 amountBNB = address(this).balance;
    payable(msg.sender).transfer((amountBNB * amountPercentage) / 100);
}

ftrace | funcSig
function getBep20Tokens(address _tokenAddress, uint256 amount)
    external
    onlyOwner
{
    require(
        IERC20(_tokenAddress).balanceOf(address(this)) >= amount,
        "No Enough Tokens"
    );
    IERC20(_tokenAddress).transfer(msg.sender, amount);
}
```

❖ The owner can change all buy and sell fees



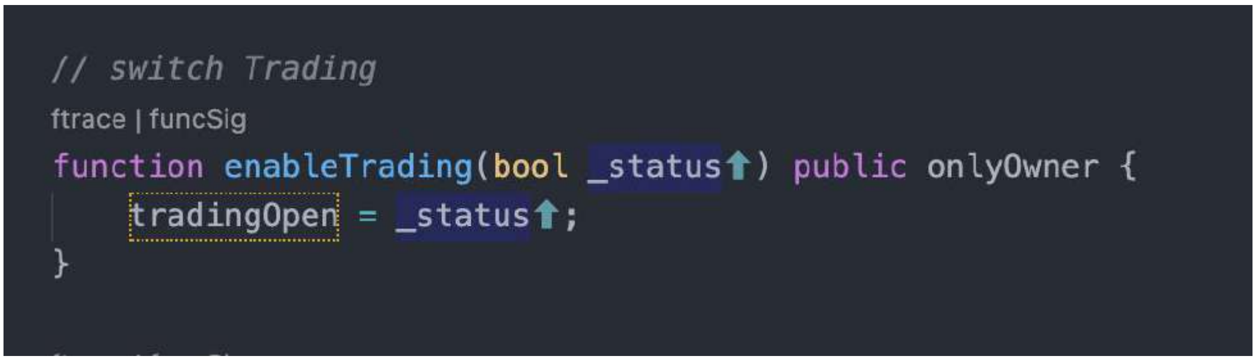
```
ftrace | funcSig
function updateBuyFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 game,
    uint256 team,
    uint256 dev
) public onlyOwner {
    buyRewardFee = reward;
    buyMarketingFee = marketing;
    buyLiquidityFee = liquidity;
    buyGameFee = game;
    buyDevFee = team;
    buyTeamFee = dev;

    buyTotalFees = reward
        .add(marketing)
        .add(liquidity)
        .add(game)
        .add(team)
        .add(dev);
}

ftrace | funcSig
function updateSellFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 game,
    uint256 team,
    uint256 dev
) public onlyOwner {
    sellRewardFee = reward;
    sellMarketingFee = marketing;
    sellLiquidityFee = liquidity;
    sellGameFee = game;
    sellDevFee = team;
    sellTeamFee = dev;

    sellTotalFees = reward
        .add(marketing)
        .add(liquidity)
        .add(game)
        .add(team)
        .add(dev);
}
```

❖ The owner can change all swap percentages



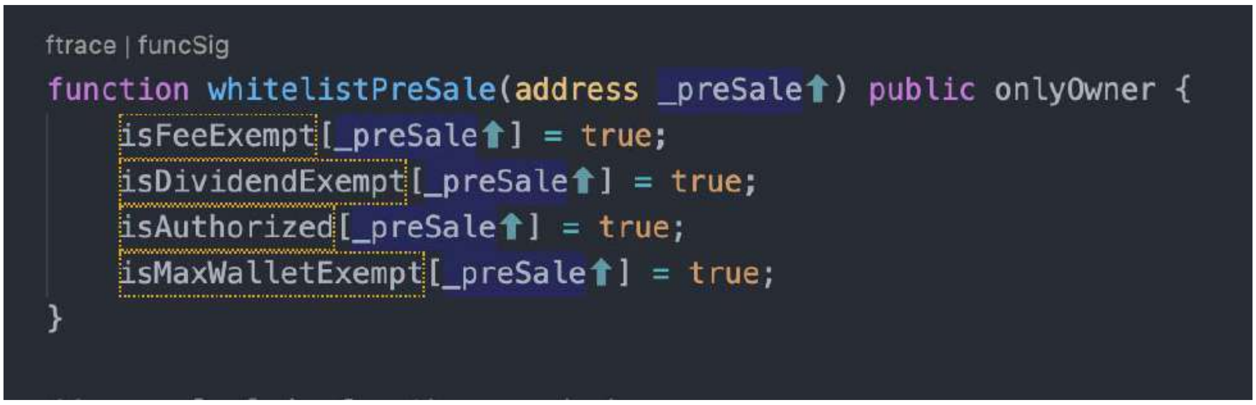
```
// update swap percentages
ftrace | funcSig
function updateSwapPercentages(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 game,
    uint256 team,
    uint256 dev
) public onlyOwner {
    rewardSwap = reward;
    marketingSwap = marketing;
    liquiditySwap = liquidity;
    gameSwap = game;
    teamSwap = team;
    devSwap = dev;
    totalSwap = reward
        .add(marketing)
        .add(liquidity)
        .add(game)
        .add(team)
        .add(dev);
}
```

❖ The owner can enable/disable trading anytime



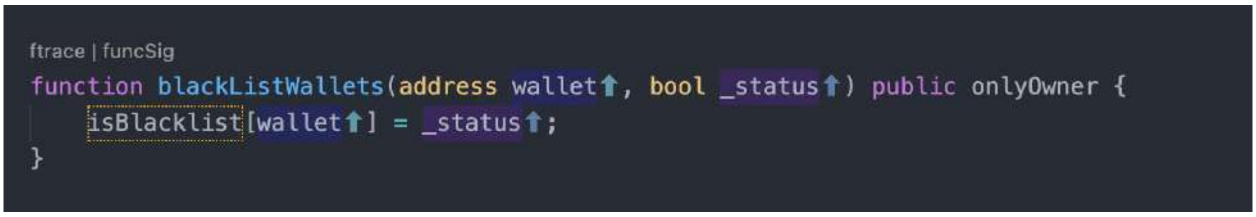
```
// switch Trading
ftrace | funcSig
function enableTrading(bool _status) public onlyOwner {
    tradingOpen = _status;
}
```

❖ The owner can whitelist presale address



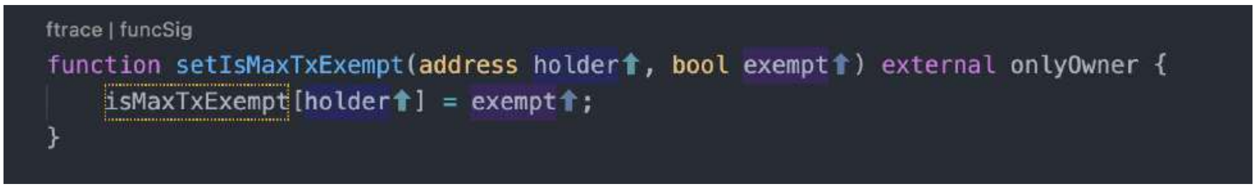
```
ftrace | funcSig
function whitelistPreSale(address _preSale) public onlyOwner {
    isFeeExempt[_preSale] = true;
    isDividendExempt[_preSale] = true;
    isAuthorized[_preSale] = true;
    isMaxWalletExempt[_preSale] = true;
}
```

❖ The owner can blacklist/unblock wallet address



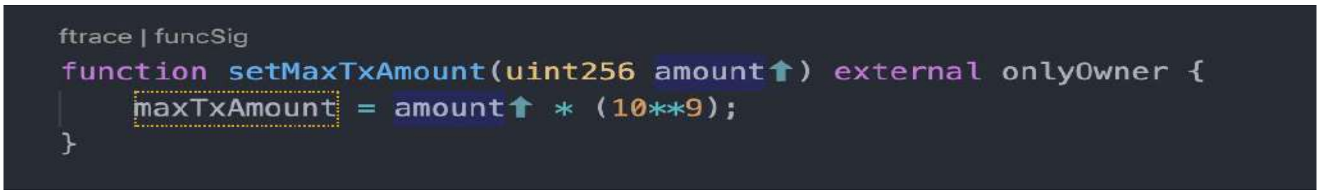
```
ftrace | funcSig
function blackListWallets(address wallet, bool _status) public onlyOwner {
    isBlacklist[wallet] = _status;
}
```

❖ The owner can include/exclude wallets from max transaction



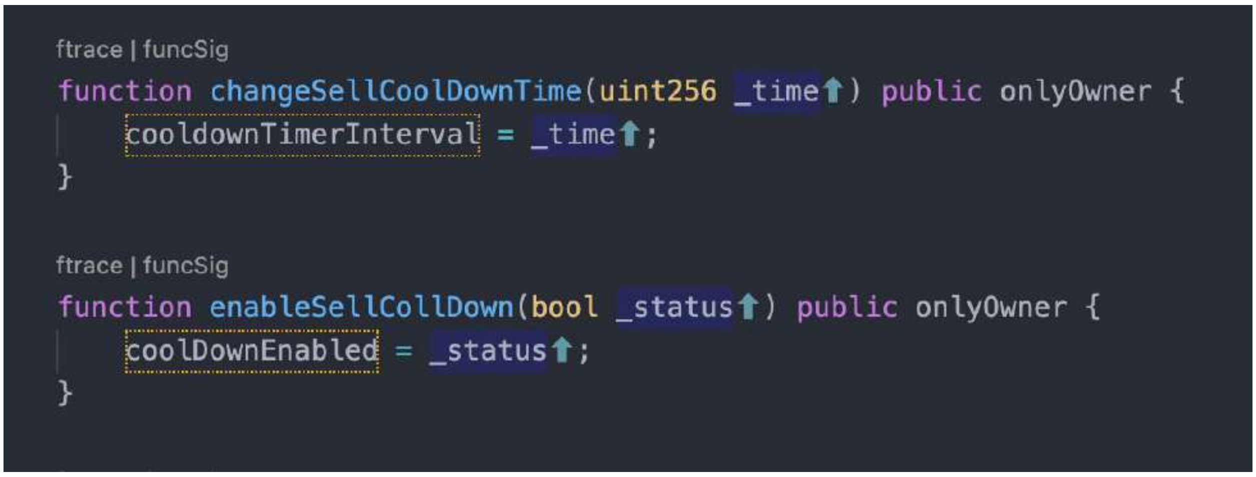
```
ftrace | funcSig
function setIsMaxTxExempt(address holder, bool exempt) external onlyOwner {
    isMaxTxExempt[holder] = exempt;
}
```

❖ The owner can change max transaction amount



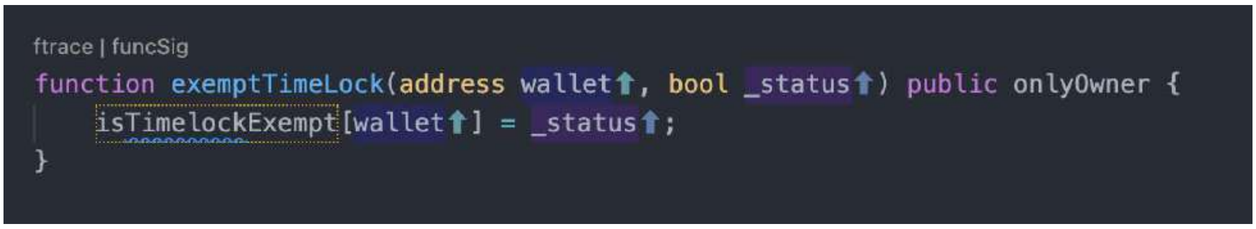
```
ftrace | funcSig
function setMaxTxAmount(uint256 amount) external onlyOwner {
    maxTxAmount = amount * (10**9);
}
```

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



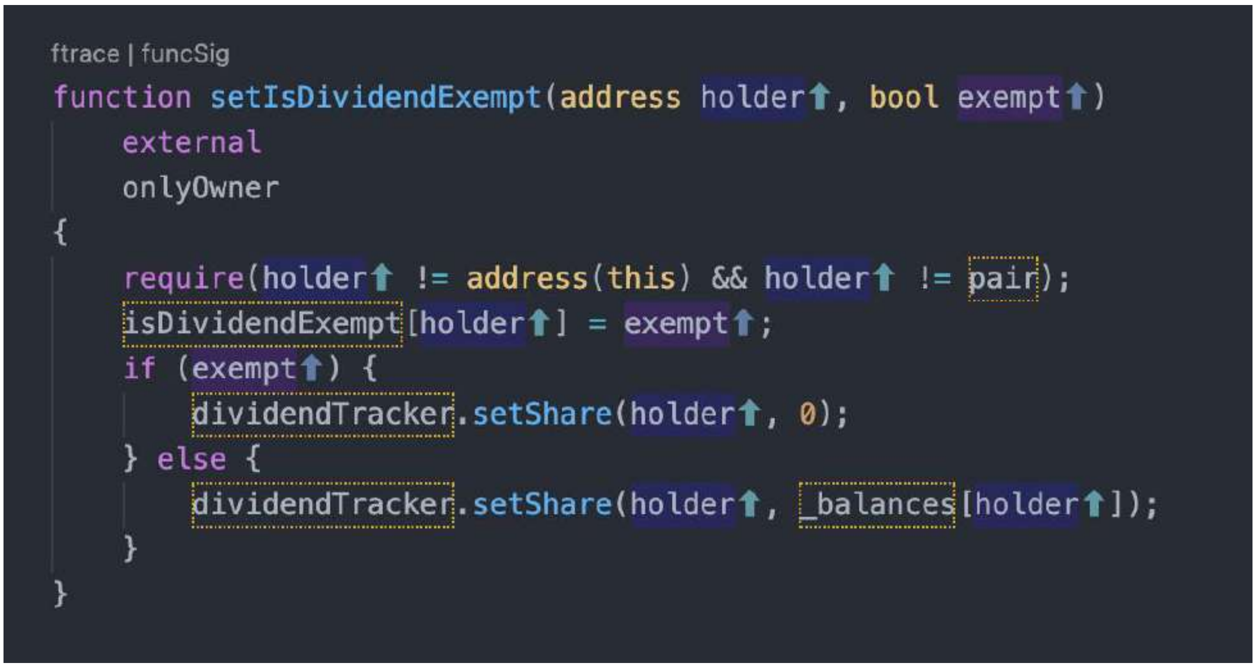
```
ftrace | funcSig
function changeSellCoolDownTime(uint256 _time) public onlyOwner {
    cooldownTimerInterval = _time;
}

ftrace | funcSig
function enableSellCollDown(bool _status) public onlyOwner {
    coolDownEnabled = _status;
}
```

❖ The owner can include/exclude wallets from sell cool down



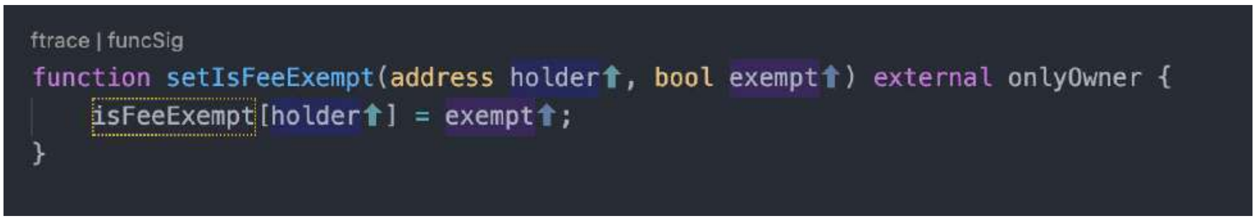
```
ftrace | funcSig
function exemptTimeLock(address wallet, bool _status) public onlyOwner {
    isTimelockExempt[wallet] = _status;
}
```

❖ The owner can include/exclude wallets from dividend

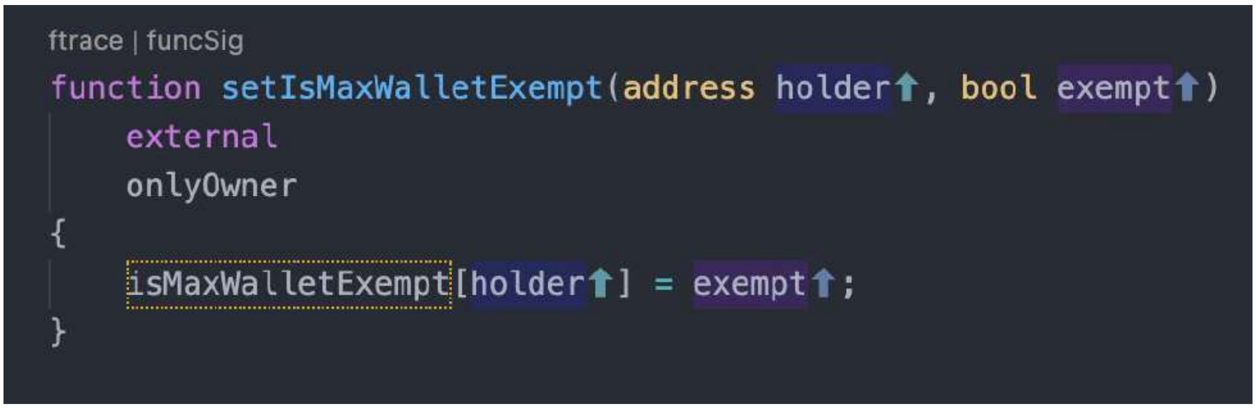


```
ftrace | funcSig
function setIsDividendExempt(address holder, bool exempt)
    external
    onlyOwner
{
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if (exempt) {
        dividendTracker.setShare(holder, 0);
    } else {
        dividendTracker.setShare(holder, _balances[holder]);
    }
}
```

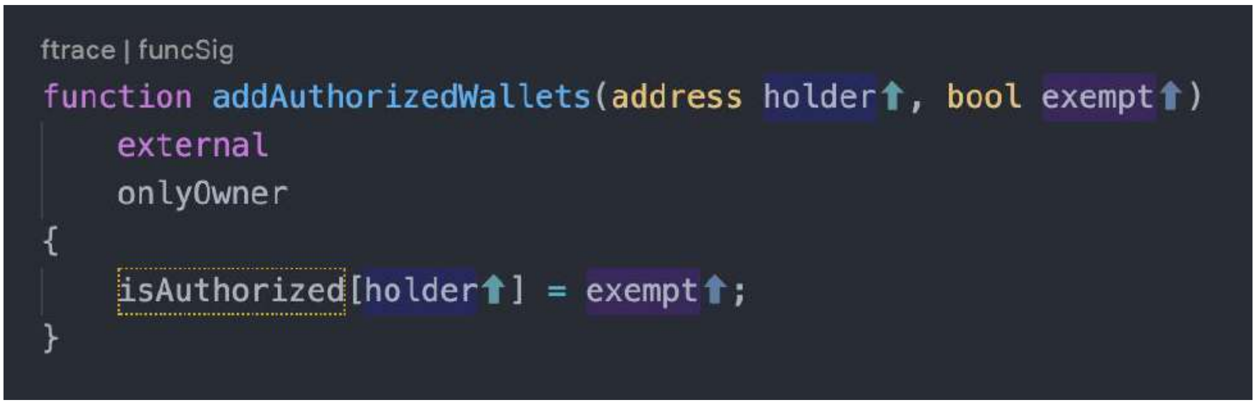
❖ The owner can include/exclude wallets from fees


```
ftrace | funcSig
function setIsFeeExempt(address holder, bool exempt) external onlyOwner {
    isFeeExempt[holder] = exempt;
}
```

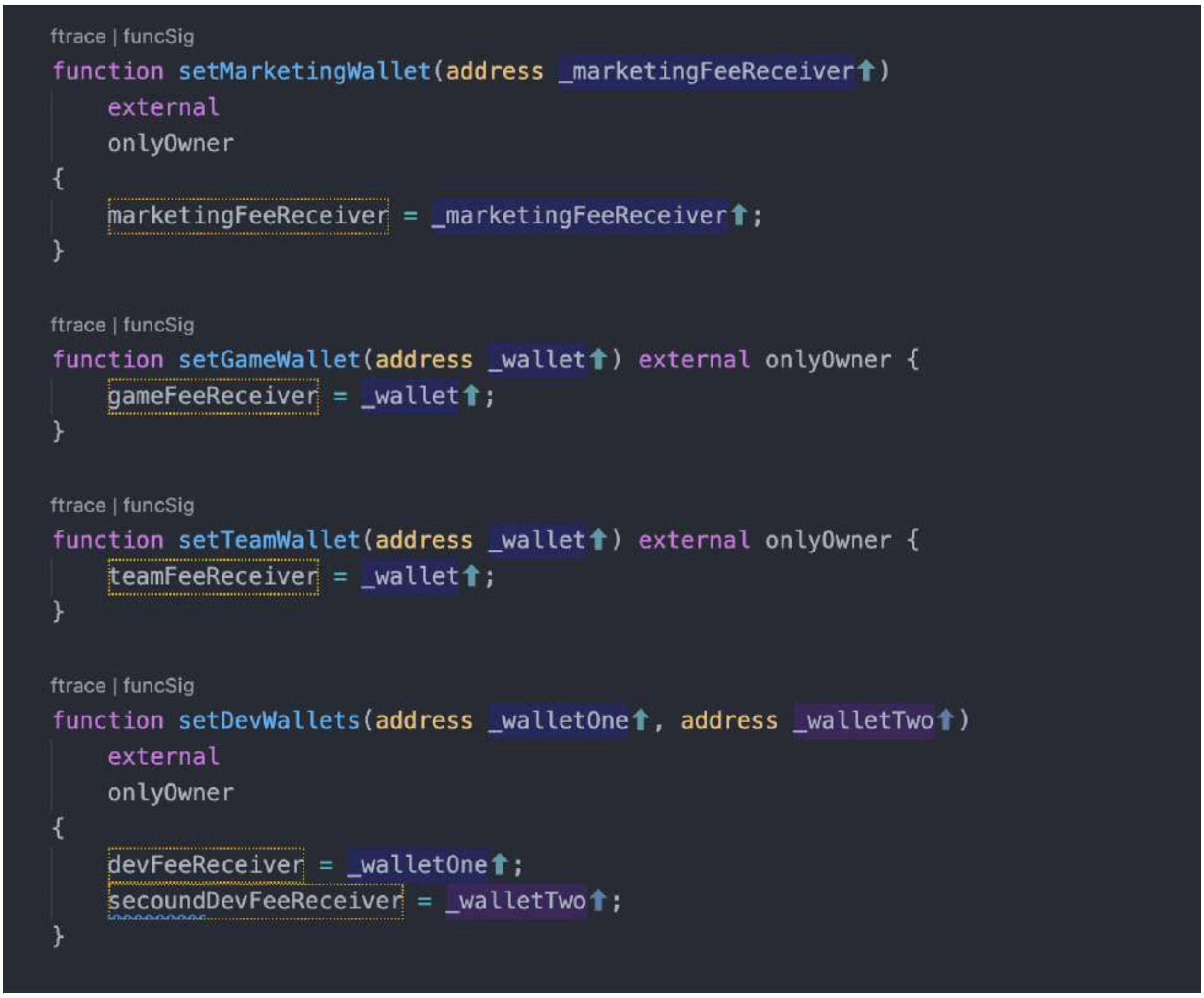
❖ The owner can include/exclude wallets from max wallet tokens


```
ftrace | funcSig
function setIsMaxWalletExempt(address holder, bool exempt)
    external
    onlyOwner
{
    isMaxWalletExempt[holder] = exempt;
}
```

❖ The owner can include/exclude wallets from authorize (authorize wallets can do transactions when trading is disabled)

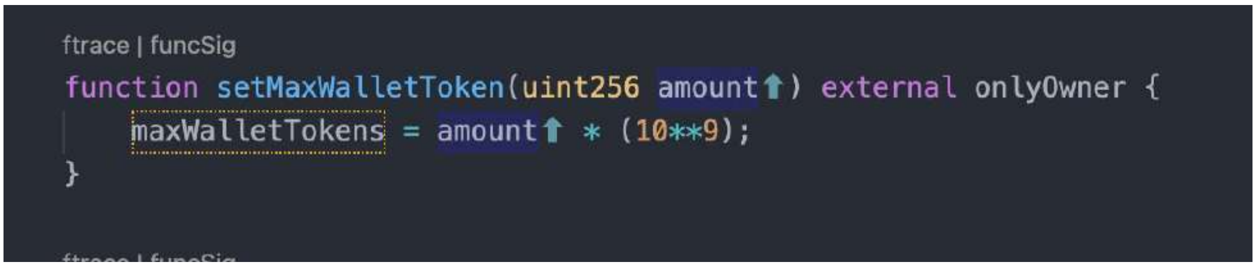

```
ftrace | funcSig
function addAuthorizedWallets(address holder, bool exempt)
    external
    onlyOwner
{
    isAuthorized[holder] = exempt;
}
```

❖ The owner can change all fees receiver wallets



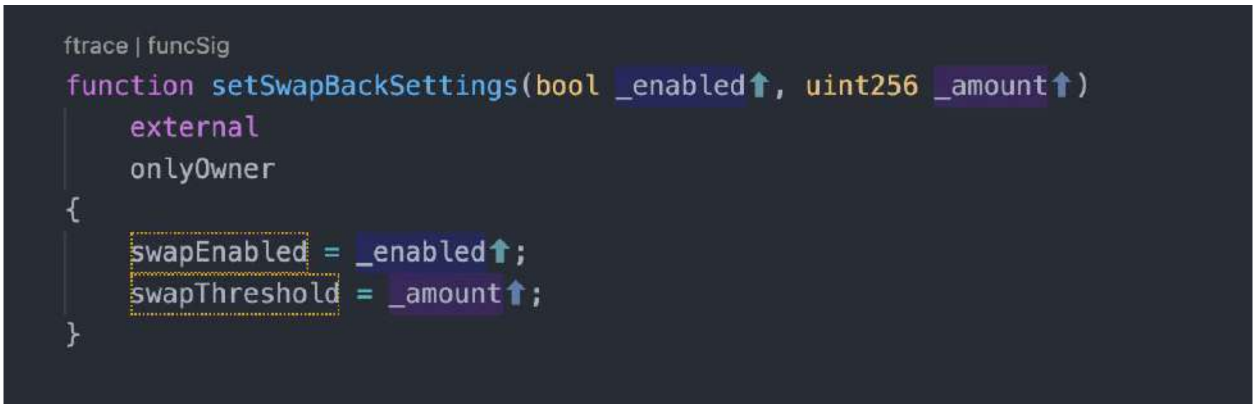
```
ftrace | funcSig
function setMarketingWallet(address _marketingFeeReceiver)
    external
    onlyOwner
{
    marketingFeeReceiver = _marketingFeeReceiver;
}

ftrace | funcSig
function setGameWallet(address _wallet) external onlyOwner {
    gameFeeReceiver = _wallet;
}

ftrace | funcSig
function setTeamWallet(address _wallet) external onlyOwner {
    teamFeeReceiver = _wallet;
}

ftrace | funcSig
function setDevWallets(address _walletOne, address _walletTwo)
    external
    onlyOwner
{
    devFeeReceiver = _walletOne;
    secoundDevFeeReceiver = _walletTwo;
}
```

❖ The owner can change max wallet tokens



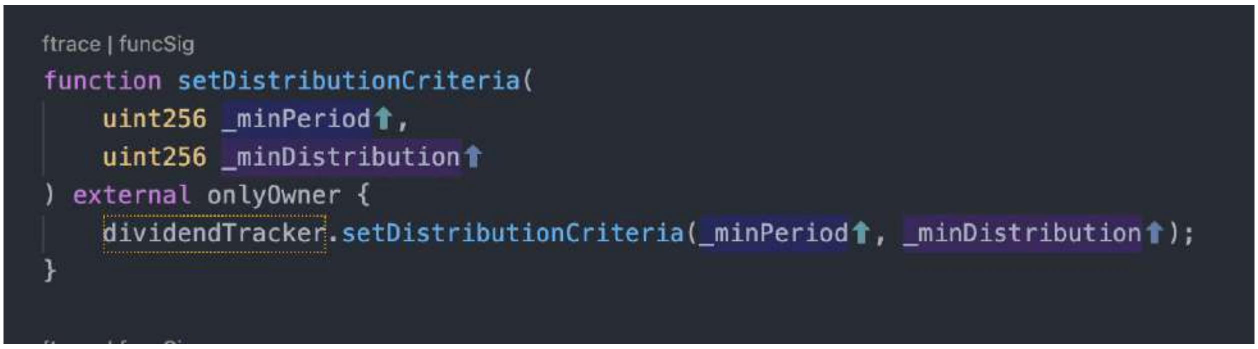
```
ftrace | funcSig
function setMaxWalletToken(uint256 amount) external onlyOwner {
    maxWalletTokens = amount * (10**9);
}
```

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



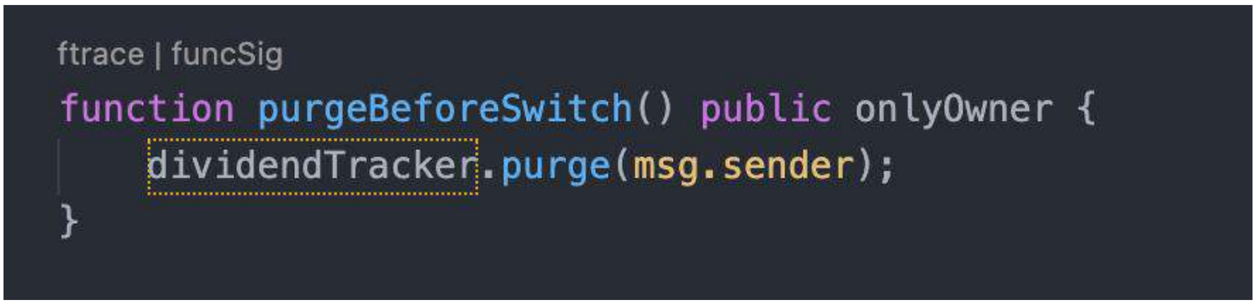
```
ftrace | funcSig
function setSwapBackSettings(bool _enabled, uint256 _amount)
    external
    onlyOwner
{
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

- The owner can change the minimum distribute token amount and minimum distribute time in rewards



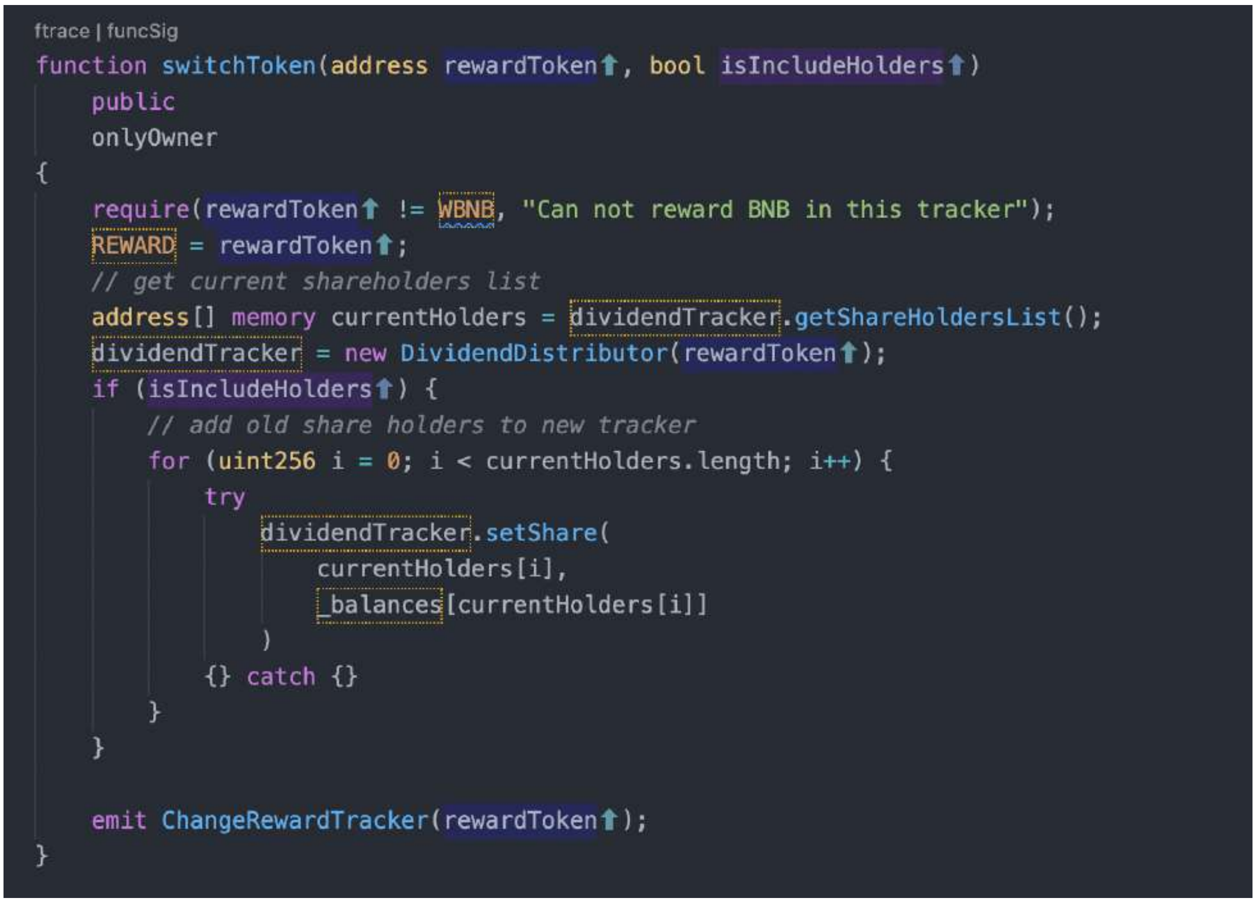
```
ftrace | funcSig
function setDistributionCriteria(
    uint256 _minPeriod,
    uint256 _minDistribution
) external onlyOwner {
    dividendTracker.setDistributionCriteria(_minPeriod, _minDistribution);
}
```

- The owner can get all tokens in the reward tracker to the owner wallet (this will have to use before change reward token)



```
ftrace | funcSig
function purgeBeforeSwitch() public onlyOwner {
    dividendTracker.purge(msg.sender);
}
```

❖ The owner can change the reward token address



```
ftrace | funcSig
function switchToken(address rewardToken, bool isIncludeHolders)
    public
    onlyOwner
{
    require(rewardToken != WBNB, "Can not reward BNB in this tracker");
    REWARD = rewardToken;
    // get current shareholders list
    address[] memory currentHolders = dividendTracker.getShareHoldersList();
    dividendTracker = new DividendDistributor(rewardToken);
    if (isIncludeHolders) {
        // add old share holders to new tracker
        for (uint256 i = 0; i < currentHolders.length; i++) {
            try
                dividendTracker.setShare(
                    currentHolders[i],
                    _balances[currentHolders[i]]
                )
            {} catch {}
        }
    }

    emit ChangeRewardTracker(rewardToken);
}
```

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: **3**

Solidity code functional issue level: **PASSED**

Number of owner privileges: **23**

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

Smart contract active ownership: **YES**