RugFreeCoins Audit

Captain HODL Token Audit

Smart Contract Security Audit

September 3, 2021

Contents

Audit details

Audited project
Captain HODL Token

Contract Address
0x2e1a49d4A9f4Fa11Fa462c49dCD8Fdf14d41EadD

Client contact
Captain HODL Team

Blockchain
Binance smart chain

Project website
captainhodl.org

Disclaimer

Background

Rugfreecoins was commissioned by Captain HODL to perform an audit of the smart contract.

https://bscscan.com/token/0x2e1a49d4A9f4Fa11Fa462c49dCD8Fdf14d41EadD

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 improve the security posture of the smart contract by remediating the issues that were identified.

About the project

Captain HODL is a token built on the Binance Smart Chain. Each transaction, purchase incur a 9% fee, and sales incur a 25% fee.

Features

- The main marketing strategy of Captain HODL is it collects a transaction tax that can then be used to strategically buy back the tokens sold. This smart buyback feature allows the team to print green candles on the chart at the most critical times, increasing the token's price through buybacks and burns and investor confidence through an upward trending chart. It will be exchanging 3% of the tax when buying and 18% of the tax when selling for Bnb buys back from the supply every minute and burn all tokens bought automatically.

- **4%** of each transaction when buying and selling is split amongst all holders in tokens. The holders will be eligible to receive tokens everyone hour and rewards are proportional to how many tokens each individual holds.

- The **sustainability fee of 2% marketing when buying and 3% when selling** is what allows Captain HODL to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to a marketing wallet per transaction. This way, Captain HODL will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

Tokenomics

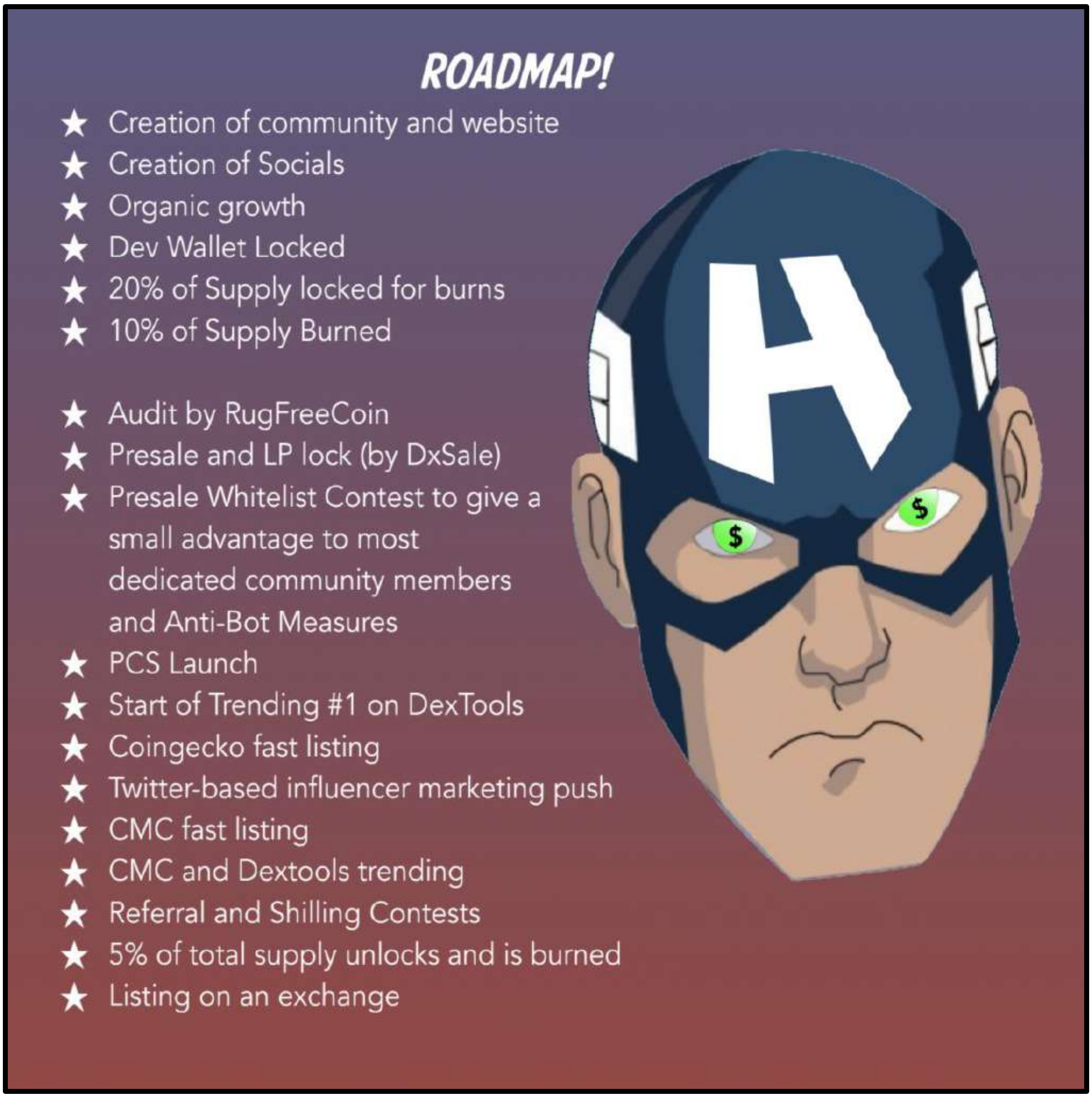
9% fee when buying

- 4% of trade goes to holders' pockets in tokens.
- 3% of trade goes to buyback & burn.
- 2% of trade goes to marketing and R&D.

25% fee when selling

- 4% of trade goes to holders’ pockets in tokens.
- 18% of trade goes to buyback & burn.
- 3% of trade goes to marketing and R&D.

Roadmap



ROADMAP!

- ★ Creation of community and website
- ★ Creation of Socials
- ★ Organic growth
- ★ Dev Wallet Locked
- ★ 20% of Supply locked for burns
- ★ 10% of Supply Burned

- ★ Audit by RugFreeCoin
- ★ Presale and LP lock (by DxSale)
- ★ Presale Whitelist Contest to give a small advantage to most dedicated community members and Anti-Bot Measures
- ★ PCS Launch
- ★ Start of Trending #1 on DexTools
- ★ Coingecko fast listing
- ★ Twitter-based influencer marketing push
- ★ CMC fast listing
- ★ CMC and Dextools trending
- ★ Referral and Shilling Contests
- ★ 5% of total supply unlocks and is burned
- ★ Listing on an exchange

Target market and the concept

Target market

- Anyone who's interested in Crypto space with long term investment plans.
- Anyone who's ready to earn a passive income in tokens by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in stake CaptainHODL tokens and earn rewards.
- Anyone who's interested in taking part with the future plans of CaptainHODL token.
- Anyone who's interested in making financial transactions with any other party using CaptainHODL as the currency.
- Anyone who's interested in taking part with the project's marketing activities and get paid in BNBs.

Core concept

The buyback & burn mechanism

The buyback and burn mechanism collect 3% tax on each transaction when buying and 18% when 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.

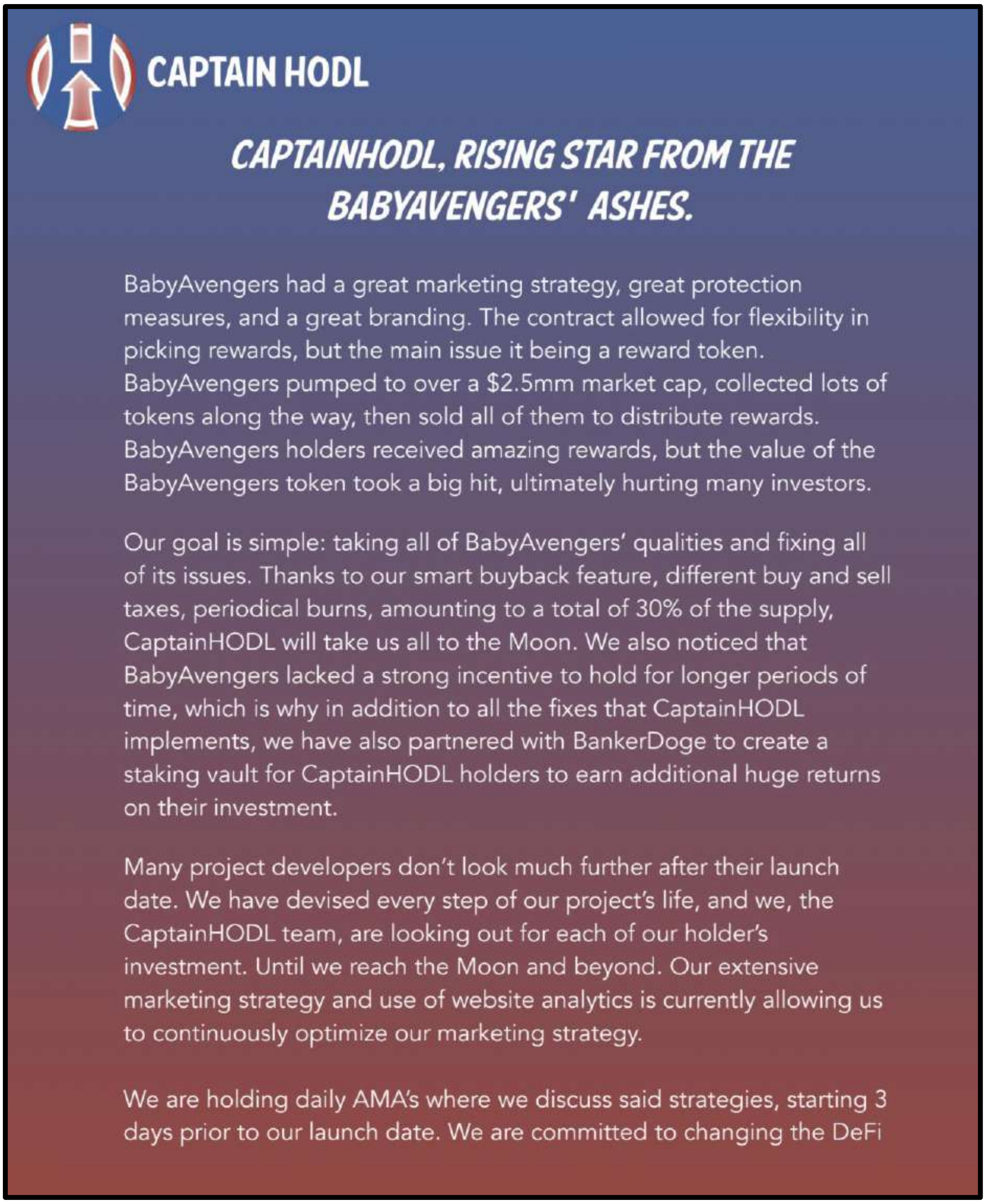
The BNB reward system

4% of each transaction gets sent in tokens and is split amongst all holders. The rewards are sent to holders in CaptainHODL tokens, holders will be eligible to receive tokens proportional to how many tokens each individual holds.

Sustainable mechanism

The fee of 2% marketing when buying and 3% marketing when selling is what allows CaptainHODL to promote the token and use funds to further development of the platform. Tokens will be swapped into BNB and will be sent to a marketing wallet per transaction. This way, CaptainHODL 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 future plan



CAPTAIN HODL

CAPTAINHODL, RISING STAR FROM THE BABYAVENGERS' ASHES.

BabyAvengers had a great marketing strategy, great protection measures, and a great branding. The contract allowed for flexibility in picking rewards, but the main issue it being a reward token. BabyAvengers pumped to over a $2.5mm market cap, collected lots of tokens along the way, then sold all of them to distribute rewards. BabyAvengers holders received amazing rewards, but the value of the BabyAvengers token took a big hit, ultimately hurting many investors.

Our goal is simple: taking all of BabyAvengers' qualities and fixing all of its issues. Thanks to our smart buyback feature, different buy and sell taxes, periodical burns, amounting to a total of 30% of the supply, CaptainHODL will take us all to the Moon. We also noticed that BabyAvengers lacked a strong incentive to hold for longer periods of time, which is why in addition to all the fixes that CaptainHODL implements, we have also partnered with BankerDoge to create a staking vault for CaptainHODL holders to earn additional huge returns on their investment.

Many project developers don't look much further after their launch date. We have devised every step of our project's life, and we, the CaptainHODL team, are looking out for each of our holder's investment. Until we reach the Moon and beyond. Our extensive marketing strategy and use of website analytics is currently allowing us to continuously optimize our marketing strategy.

We are holding daily AMA's where we discuss said strategies, starting 3 days prior to our launch date. We are committed to changing the DeFi

Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	9/10
3	Information quality	10/10
4	Service quality	9/10
5	System quality	9/10
6	Impact on the community	10/10
7	Impact on the business	10/10
8	Preparing for the future	9/10
Total Points		**9.38/10**

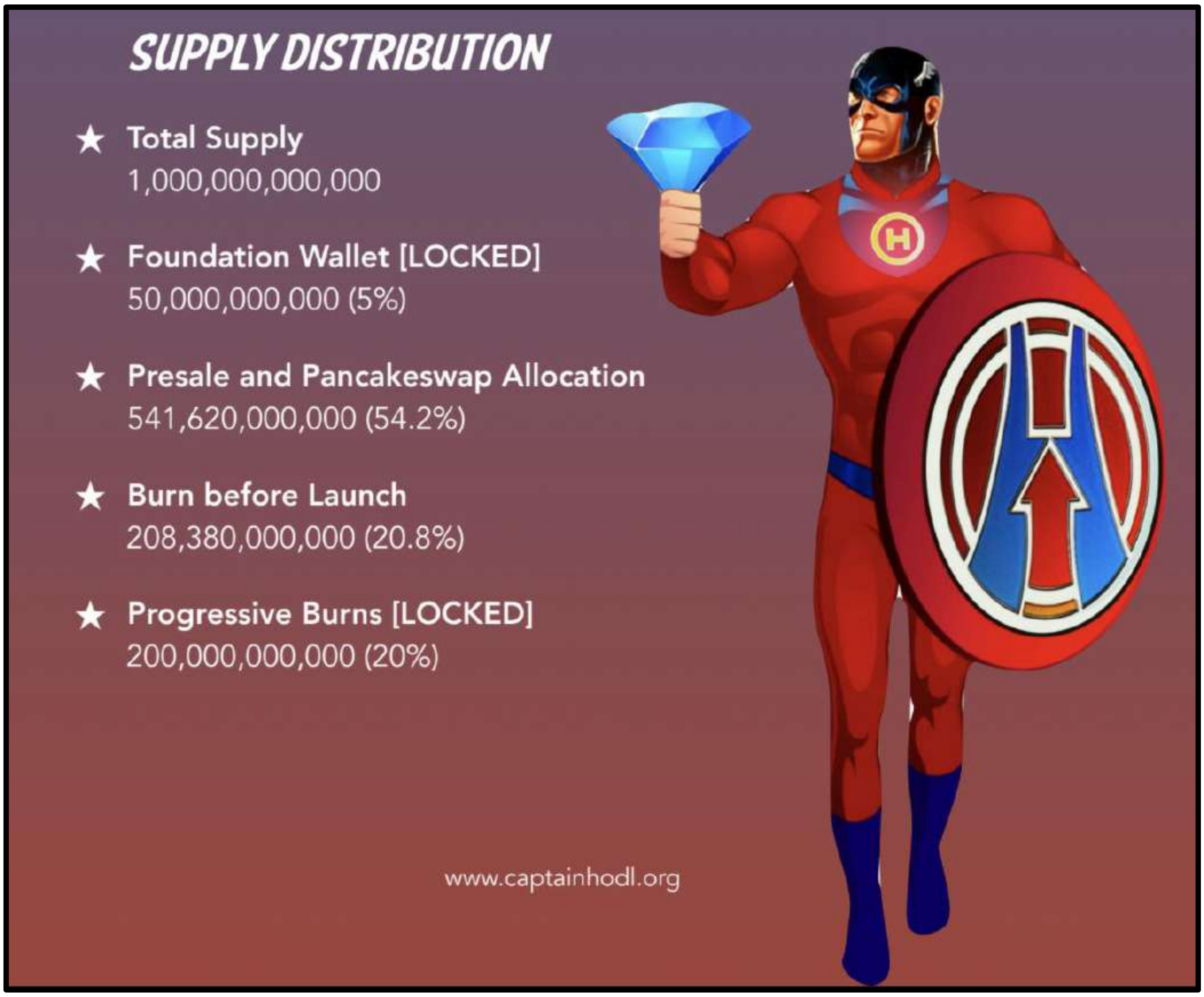
Contract details

Token contract details for 03rd September 2021

Contract name	CaptainHODL
Contract address	0x2e1a49d4A9f4Fa11Fa462c49dCD8Fdf14d41EadD
Token supply	1,000,000,000,000
Token ticker	CAPHODL
Decimals	4
Token holders	3
Transaction count	9
Marketing wallet address	not public
Contract deployer address	0x6Aa2e01308542721Ea7C5B866083dD5199df559d
Contract's current owner address	0x6aa2e01308542721ea7c5b866083dd5199df559d

Token distribution

Tokens are distributed as follows:


SUPPLY DISTRIBUTION

★ Total Supply
1,000,000,000,000

★ Foundation Wallet [LOCKED]
50,000,000,000 (5%)

★ Presale and Pancakeswap Allocation
541,620,000,000 (54.2%)

★ Burn before Launch
208,380,000,000 (20.8%)

★ Progressive Burns [LOCKED]
200,000,000,000 (20%)

www.captainhodl.org

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	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

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 its visibility and mutability.

Contract	**Type**	**Bases**		
└	Function Name	Visibility	Mutability	Modifiers
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	**Interface**			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		

Address	**Library**			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
Ownable	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	getUnlockTime	Public ❗		NO❗
└	getTime	Public ❗		NO❗
└	lock	Public ❗	🛑	onlyOwner
└	unlock	Public ❗	🛑	NO❗
IUniswapV2Factory	**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❗

IUniswapV2Pair	**Interface**			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🛑	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	burn	External ❗	🛑	NO❗
└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗

IUniswapV2Router01	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**	**IUniswapV2 Router01**		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗

CaptainHODL	**Implementation**	**Context, IERC20, Ownable**		
└		Public ❗️	🛑	NO❗️
└	name	Public ❗️		NO❗️
└	symbol	Public ❗️		NO❗️
└	decimals	Public ❗️		NO❗️
└	totalSupply	Public ❗️		NO❗️
└	balanceOf	Public ❗️		NO❗️
└	transfer	Public ❗️	🛑	NO❗️
└	allowance	Public ❗️		NO❗️
└	approve	Public ❗️	🛑	NO❗️
└	transferFrom	Public ❗️	🛑	NO❗️
└	increaseAllowance	Public ❗️	🛑	NO❗️
└	decreaseAllowance	Public ❗️	🛑	NO❗️
└	isExcludedFromReward	Public ❗️		NO❗️
└	totalFees	Public ❗️		NO❗️
└	minimumTokensBeforeSwapAmount	Public ❗️		NO❗️
└	buyBackSellLimitAmount	Public ❗️		NO❗️
└	deliver	Public ❗️	🛑	NO❗️
└	reflectionFromToken	Public ❗️		NO❗️
└	tokenFromReflection	Public ❗️		NO❗️
└	excludeFromReward	Public ❗️	🛑	onlyOwner
└	includeInReward	External ❗️	🛑	onlyOwner
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	swapTokens	Private 🔐	🛑	lockTheSwap
└	buyBackTokens	Private 🔐	🛑	lockTheSwap

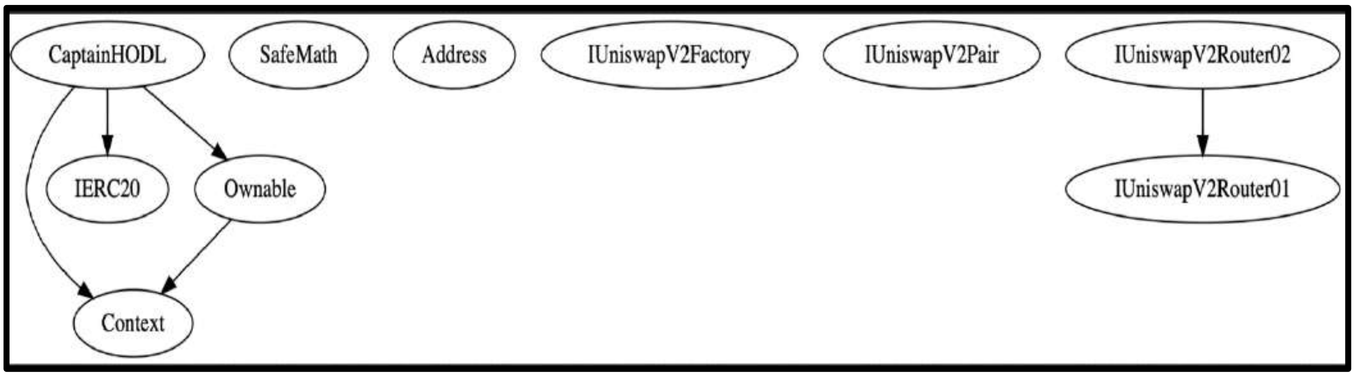
└	swapTokensForEth	Private 🔐	🛑	
└	swapETHForTokens	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	_transferStandard	Private 🔐	🛑	
└	_transferToExcluded	Private 🔐	🛑	
└	_transferFromExcluded	Private 🔐	🛑	
└	_transferBothExcluded	Private 🔐	🛑	
└	_reflectFee	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_takeLiquidity	Private 🔐	🛑	
└	calculateTaxFee	Private 🔐		
└	calculateLiquidityFee	Private 🔐		
└	removeAllFee	Private 🔐	🛑	
└	restoreAllFee	Private 🔐	🛑	
└	isExcludedFromFee	Public ❗		NO❗
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	_getSellBnBAmount	Private 🔐		
└	_removeOldSellHistories	Private 🔐	🛑	
└	SetBuyBackMaxTimeForHistories	External ❗	🛑	onlyOwner
└	SetBuyBackDivisor	External ❗	🛑	onlyOwner
└	GetBuyBackTimeInterval	Public ❗		NO❗
└	SetBuyBackTimeInterval	External ❗	🛑	onlyOwner

└	SetBuyBackRangeRate	External ❗️	🛑	onlyOwner
└	GetSwapMinutes	Public ❗️		NO❗️
└	SetSwapMinutes	External ❗️	🛑	onlyOwner
└	setTaxFeePercent	External ❗️	🛑	onlyOwner
└	setBuyFee	External ❗️	🛑	onlyOwner
└	setSellFee	External ❗️	🛑	onlyOwner
└	setLiquidityFeePercent	External ❗️	🛑	onlyOwner
└	setBuyBackSellLimit	External ❗️	🛑	onlyOwner
└	setMaxTxAmount	External ❗️	🛑	onlyOwner
└	setMarketingDivisor	External ❗️	🛑	onlyOwner
└	setNumTokensSellToAddToBuyBack	External ❗️	🛑	onlyOwner
└	setMarketingAddress	External ❗️	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	Public ❗️	🛑	onlyOwner
└	setBuyBackEnabled	Public ❗️	🛑	onlyOwner
└	setAutoBuyBackEnabled	Public ❗️	🛑	onlyOwner
└	prepareForPreSale	External ❗️	🛑	onlyOwner
└	afterPreSale	External ❗️	🛑	onlyOwner
└	transferToAddressETH	Private 🔐	🛑	
└	changeRouterVersion	Public ❗️	🛑	onlyOwner
└		External ❗️	💵	NO❗️
└	transferForeignToken	Public ❗️	🛑	onlyOwner
└	Sweep	External ❗️	🛑	onlyOwner
└	setAddressFee	External ❗️	🛑	onlyOwner
└	setBuyAddressFee	External ❗️	🛑	onlyOwner
└	setSellAddressFee	External ❗️	🛑	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.**

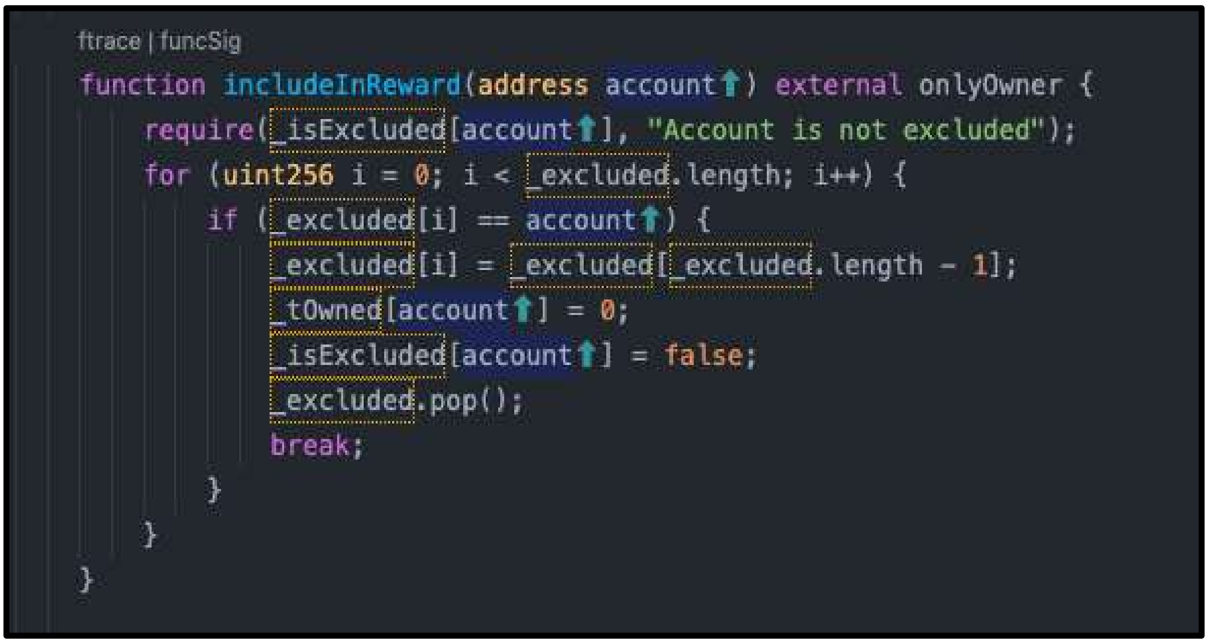
❖ **Medium severity issues**

- **No medium severity issues found.**

❖ **Low severity issues**

- **Out of gas issue**

In the includeInReward function, if they use a long wallet list there can be an OUT_OF_GAS issue, better to use a small array list at once.

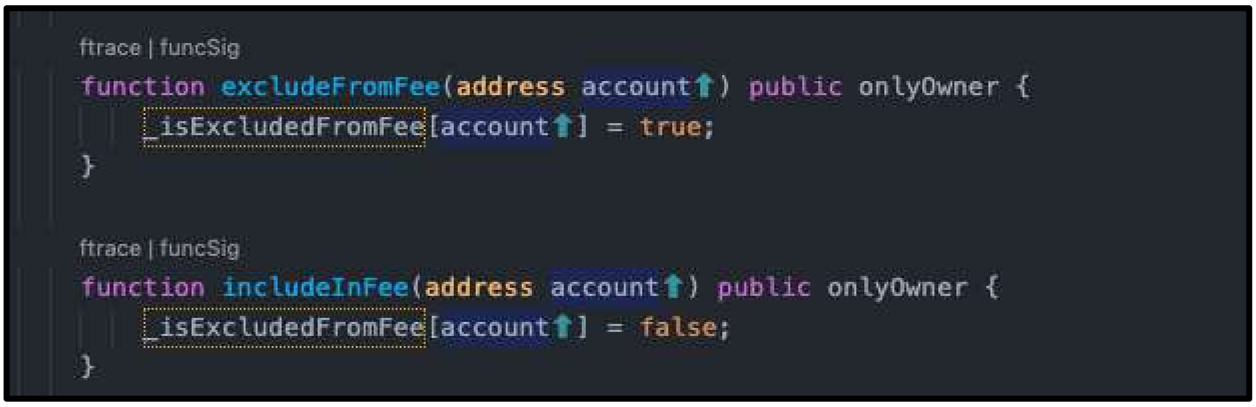


```
ftrace | funcSig
function includeInReward(address account) external onlyOwner {
    require(_isExcluded[account], "Account is not excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- **Liquidity is not getting added to the liquidity pool against all the trades, which might be a reason to lead to a market and token price imbalance. Owners can manually add liquidity from time to time but, still not a good feasible solution.**

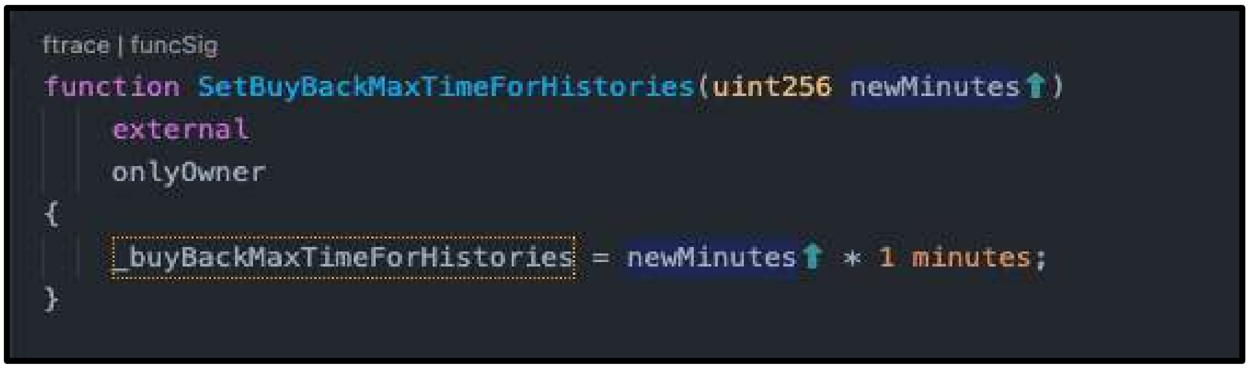
Owner privileges

❖ The owner can include/exclude wallets from fees.



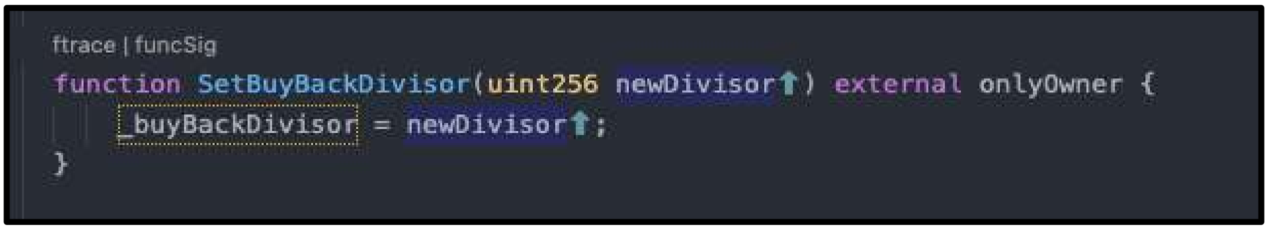
```
ftrace | funcSig
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}

ftrace | funcSig
function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}
```

❖ The owner can change max buy back time.



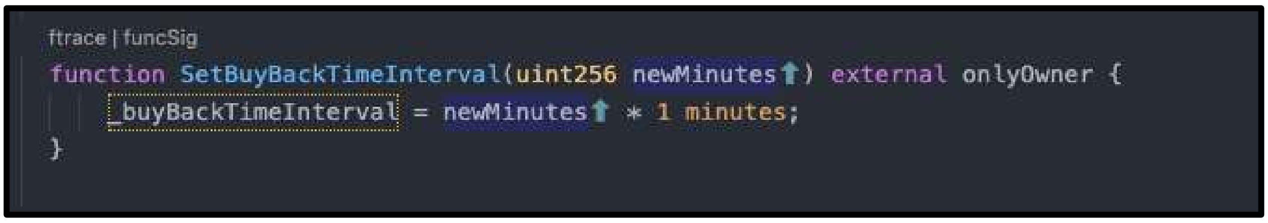
```
ftrace | funcSig
function SetBuyBackMaxTimeForHistories(uint256 newMinutes)
    external
    onlyOwner
{
    _buyBackMaxTimeForHistories = newMinutes * 1 minutes;
}
```

❖ The owner can change the buyback divisor.



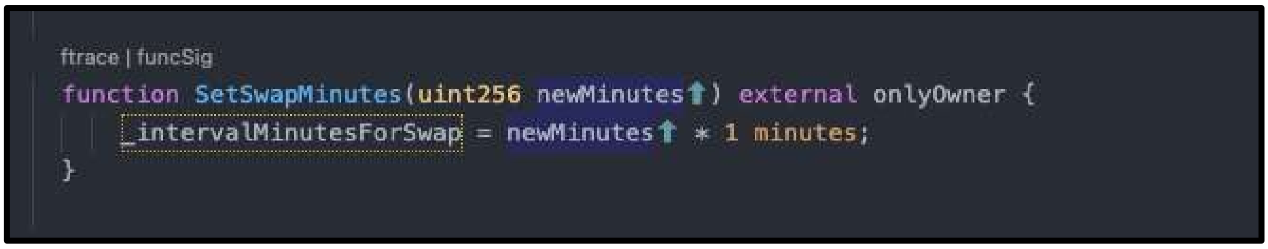
```
ftrace | funcSig
function SetBuyBackDivisor(uint256 newDivisor) external onlyOwner {
    _buyBackDivisor = newDivisor;
}
```

❖ The owner can change the buyback time interval.



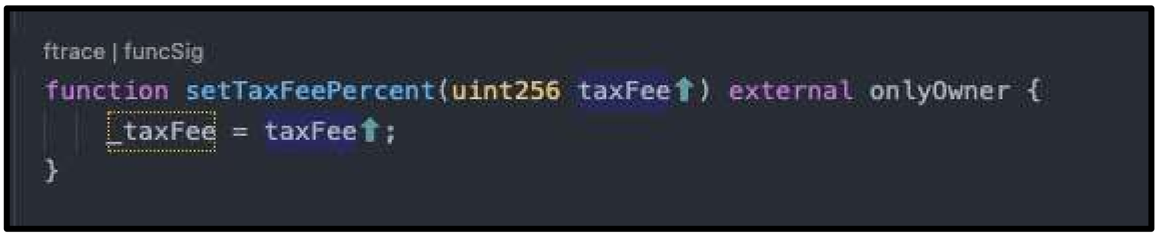
```
ftrace | funcSig
function SetBuyBackTimeInterval(uint256 newMinutes) external onlyOwner {
    _buyBackTimeInterval = newMinutes * 1 minutes;
}
```

❖ The owner can change the minimum swap time.



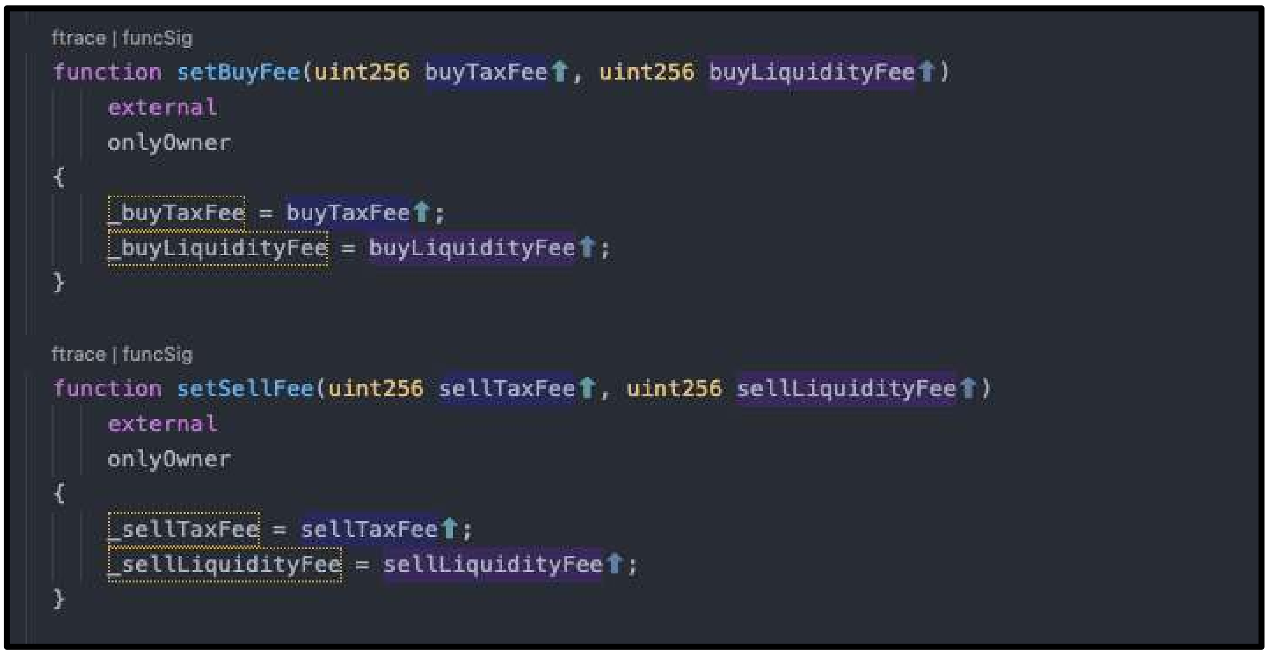
```
ftrace | funcSig
function SetSwapMinutes(uint256 newMinutes) external onlyOwner {
    _intervalMinutesForSwap = newMinutes * 1 minutes;
}
```

❖ The owner can change the normal tax fee.



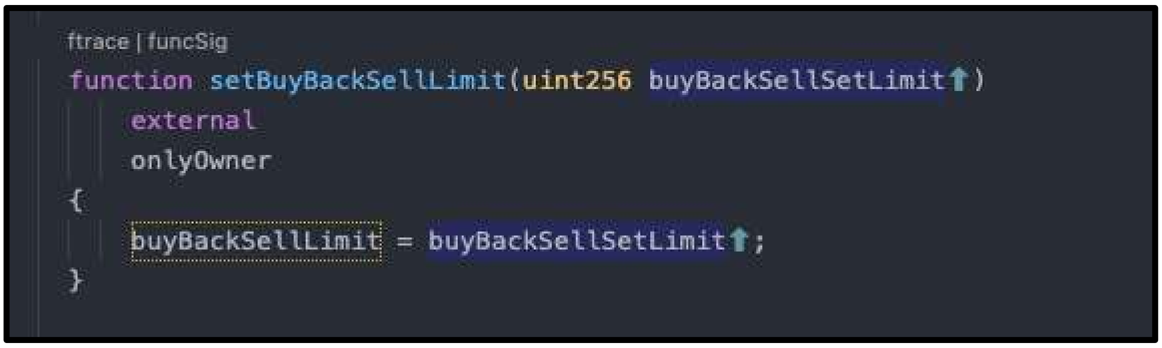
```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee) external onlyOwner {
    _taxFee = taxFee;
}
```

❖ The owner can change buy and sell fees.



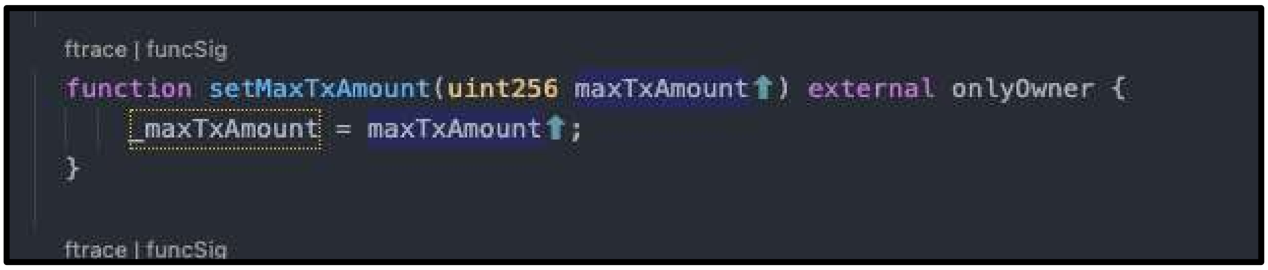
```
ftrace | funcSig
function setBuyFee(uint256 buyTaxFee, uint256 buyLiquidityFee)
    external
    onlyOwner
{
    _buyTaxFee = buyTaxFee;
    _buyLiquidityFee = buyLiquidityFee;
}

ftrace | funcSig
function setSellFee(uint256 sellTaxFee, uint256 sellLiquidityFee)
    external
    onlyOwner
{
    _sellTaxFee = sellTaxFee;
    _sellLiquidityFee = sellLiquidityFee;
}
```

❖ The owner can change the buyback sell limit.



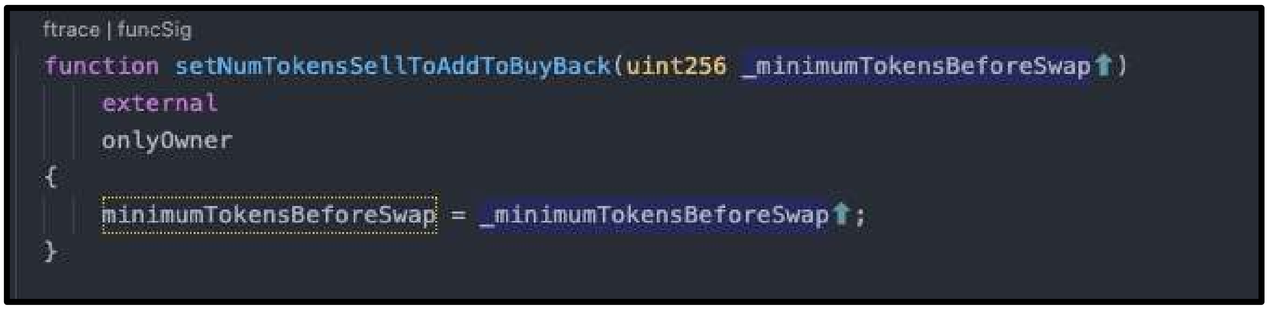
```
ftrace | funcSig
function setBuyBackSellLimit(uint256 buyBackSellSetLimit)
    external
    onlyOwner
{
    buyBackSellLimit = buyBackSellSetLimit;
}
```

❖ The owner can change the max transaction amount.



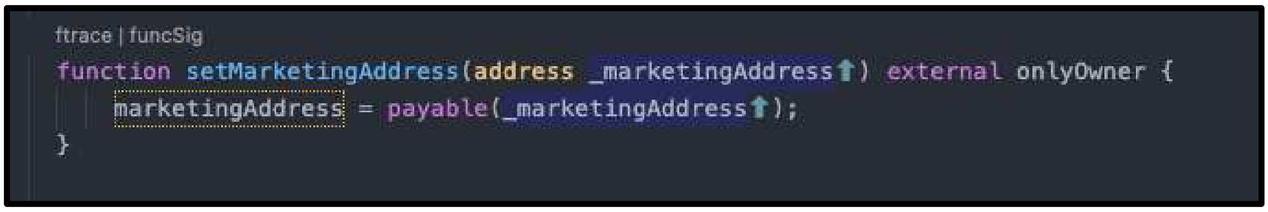
```
ftrace | funcSig
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    _maxTxAmount = maxTxAmount;
}

ftrace | funcSig
```

❖ The owner can change the number of token sells to trigger buy back.



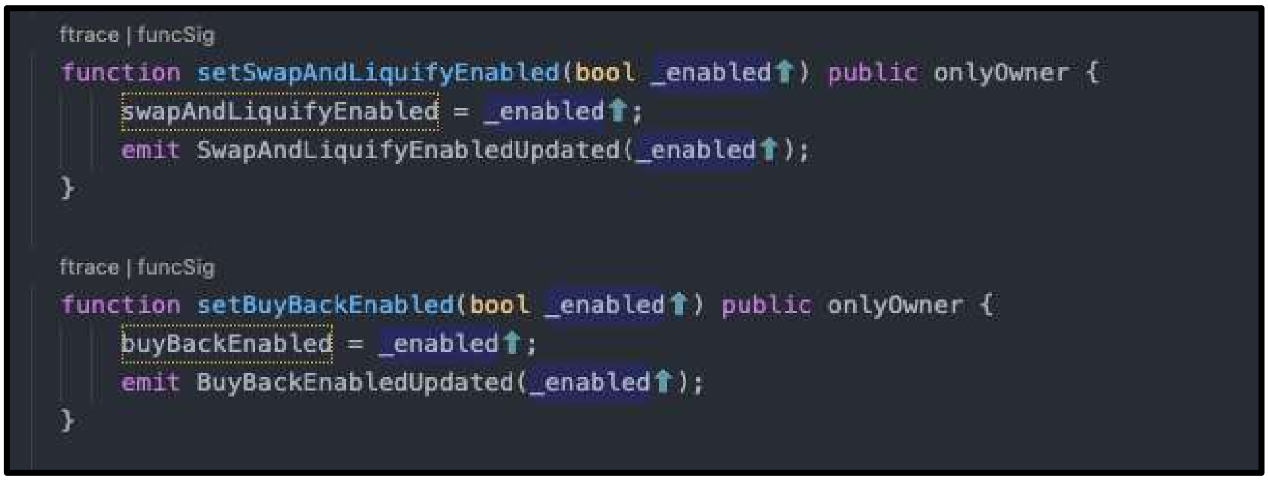
```
ftrace | funcSig
function setNumTokensSellToAddToBuyBack(uint256 _minimumTokensBeforeSwap)
    external
    onlyOwner
{
    minimumTokensBeforeSwap = _minimumTokensBeforeSwap;
}
```

❖ The owner can change the marketing address.



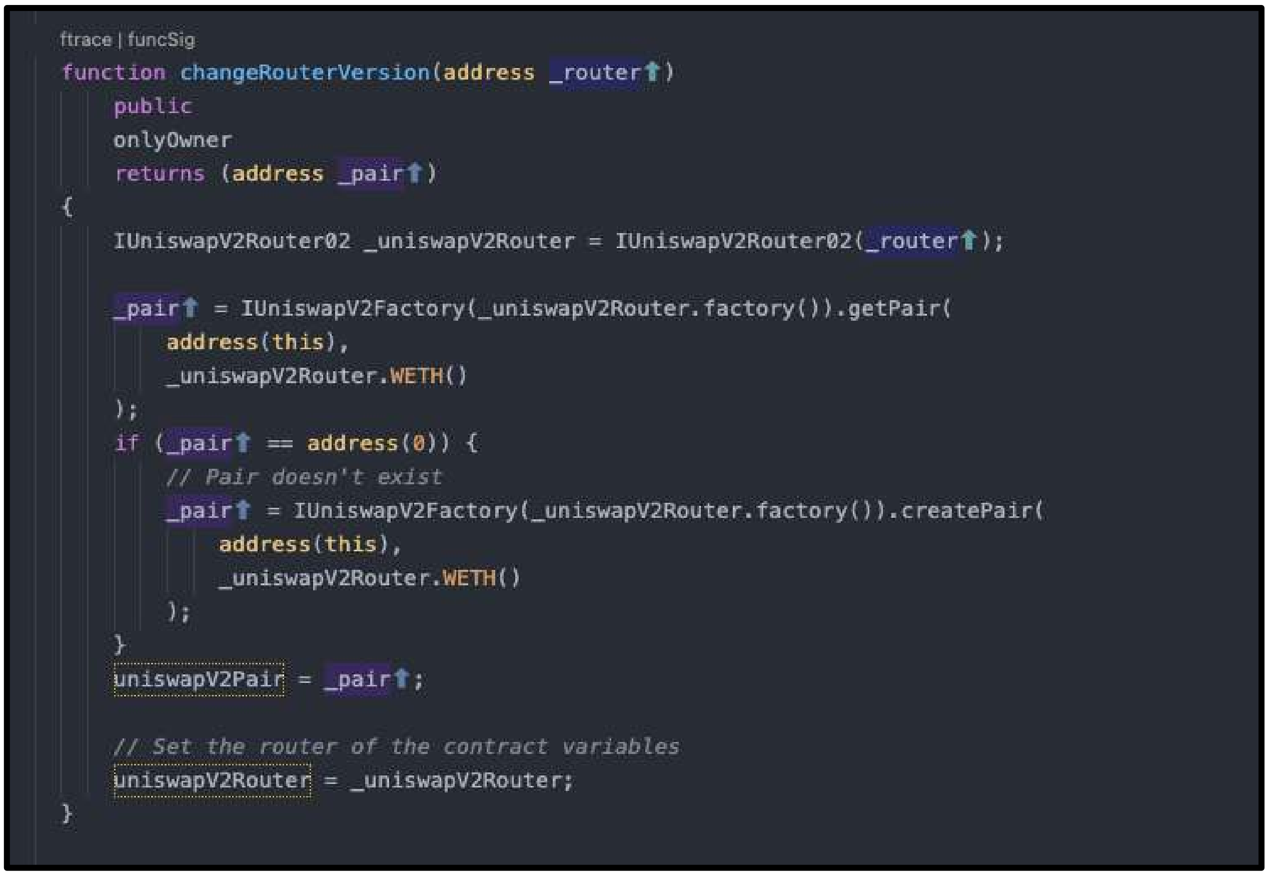
```
ftrace | funcSig
function setMarketingAddress(address _marketingAddress) external onlyOwner {
    marketingAddress = payable(_marketingAddress);
}
```

❖ The owner can enable/disable liquidity add and buy back.

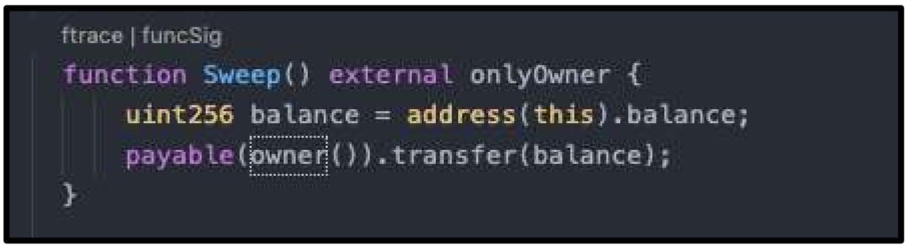


```
ftrace | funcSig
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}

ftrace | funcSig
function setBuyBackEnabled(bool _enabled) public onlyOwner {
    buyBackEnabled = _enabled;
    emit BuyBackEnabledUpdated(_enabled);
}
```

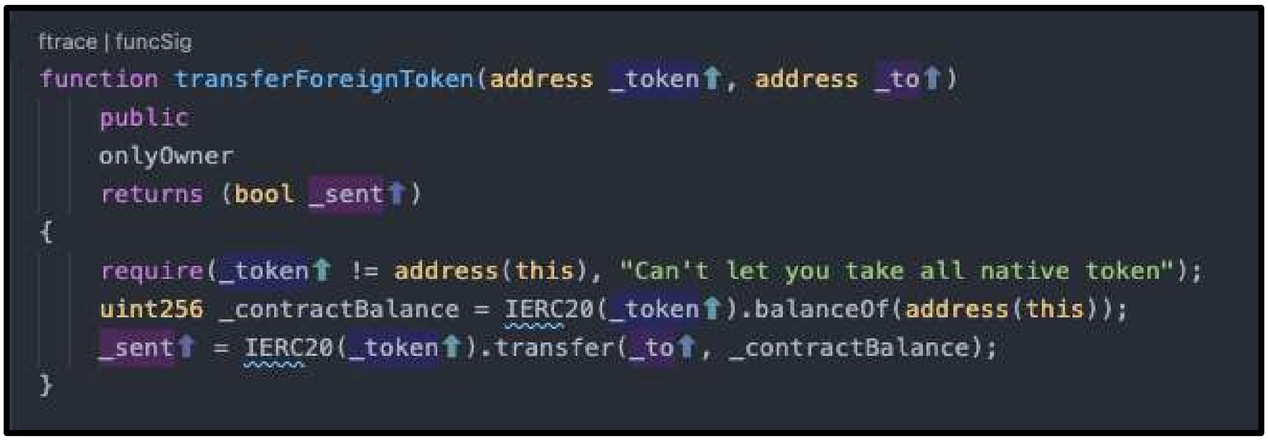
❖ The owner can change the router address.


```
ftrace | funcSig
function changeRouterVersion(address _router)
    public
    onlyOwner
    returns (address _pair)
{
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(_router);

    _pair = IUniswapV2Factory(_uniswapV2Router.factory()).getPair(
        address(this),
        _uniswapV2Router.WETH()
    );
    if (_pair == address(0)) {
        // Pair doesn't exist
        _pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(
            address(this),
            _uniswapV2Router.WETH()
        );
    }
    uniswapV2Pair = _pair;

    // Set the router of the contract variables
    uniswapV2Router = _uniswapV2Router;
}
```

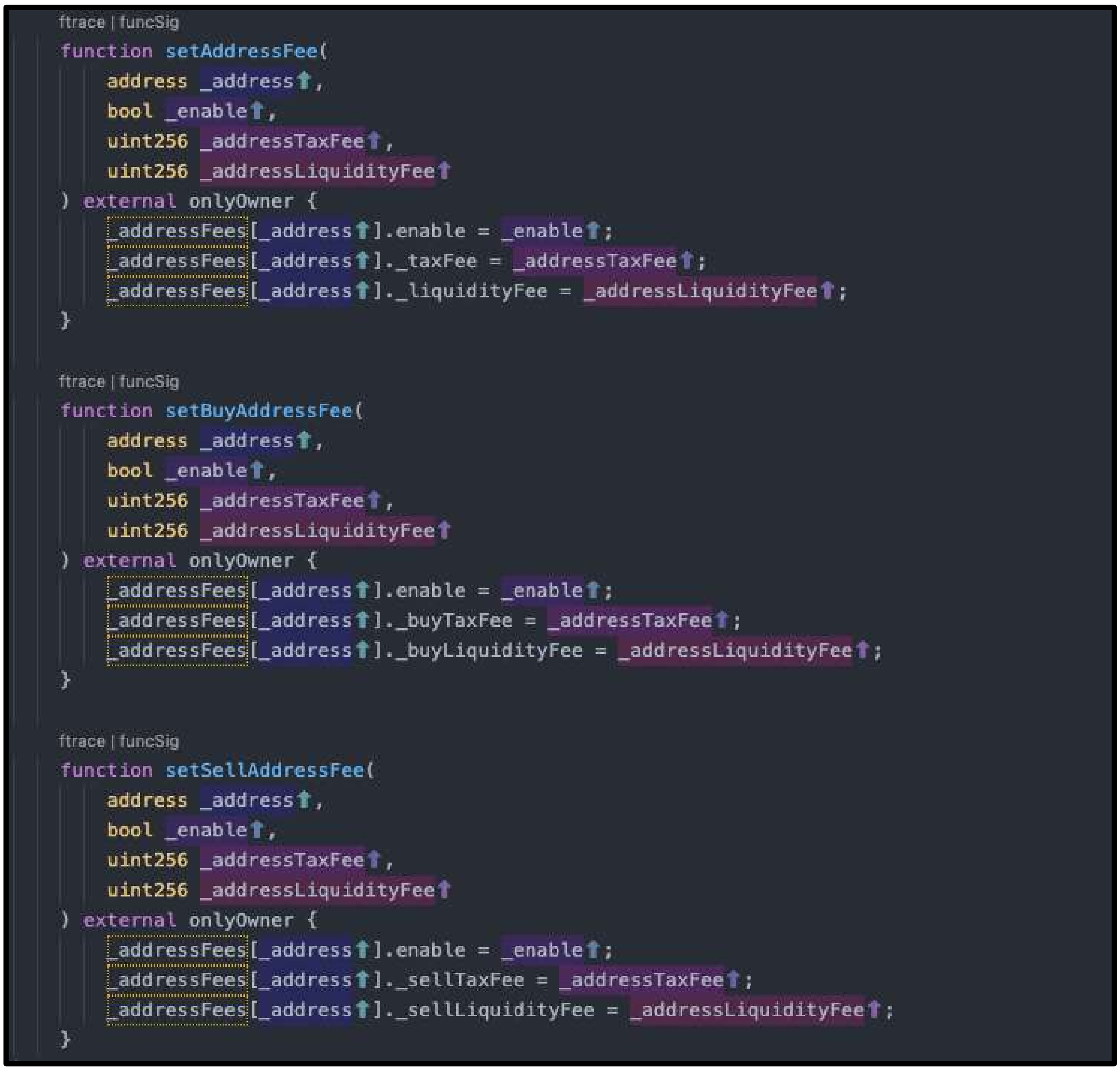
❖ The owner can transfer the wallet BNB balance to the owner account.


```
ftrace | funcSig
function Sweep() external onlyOwner {
    uint256 balance = address(this).balance;
    payable(owner()).transfer(balance);
}
```

❖ The owner can transfer any foreign tokens to any address.


```
ftrace | funcSig
function transferForeignToken(address _token, address _to)
    public
    onlyOwner
    returns (bool _sent)
{
    require(_token != address(this), "Can't let you take all native token");
    uint256 _contractBalance = IERC20(_token).balanceOf(address(this));
    _sent = IERC20(_token).transfer(_to, _contractBalance);
}
```

❖ The owner can change all fees.



```
ftrace | funcSig
function setAddressFee(
    address _address,
    bool _enable,
    uint256 _addressTaxFee,
    uint256 _addressLiquidityFee
) external onlyOwner {
    _addressFees[_address].enable = _enable;
    _addressFees[_address]._taxFee = _addressTaxFee;
    _addressFees[_address]._liquidityFee = _addressLiquidityFee;
}

ftrace | funcSig
function setBuyAddressFee(
    address _address,
    bool _enable,
    uint256 _addressTaxFee,
    uint256 _addressLiquidityFee
) external onlyOwner {
    _addressFees[_address].enable = _enable;
    _addressFees[_address]._buyTaxFee = _addressTaxFee;
    _addressFees[_address]._buyLiquidityFee = _addressLiquidityFee;
}

ftrace | funcSig
function setSellAddressFee(
    address _address,
    bool _enable,
    uint256 _addressTaxFee,
    uint256 _addressLiquidityFee
) external onlyOwner {
    _addressFees[_address].enable = _enable;
    _addressFees[_address]._sellTaxFee = _addressTaxFee;
    _addressFees[_address]._sellLiquidityFee = _addressLiquidityFee;
}
```

Audit conclusion

While conducting the audit of the CaptainHODL smart contract, it was observed that there is nothing alarming with the code and it only contains low severity issues.