RugFreeCoins Audit

NEKO ANGEL Token Audit

Smart Contract Security Audit

August 06, 2021

Contents

Audit details

Audited project

NEKO ANGEL Token

Contract Address

0x37eBB7138122DF4954d14144b3a8E75279D89E3B

Client contact

NEKO ANGEL Team

Blockchain

Binance smart chain

Project website

https://nekoangel.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 NEKO ANGEL to perform an audit of the smart contract.

https://bscscan.com/token/0x37eBB7138122DF4954d14144b3a8E75279D89E3B

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

NEKO ANGEL Token is a token built on the Binance Smart Chain and it is a community-based project. The NEKO ANGEL team devotes time and hard work to create a lovely atmosphere for every member. Each transaction incurs a 15% fee when buying and selling.

Features:

- ❖ The **automatic BNB reward of 10%** will be distributed among every holder proportional to how many tokens each individual holds.
- ❖ The liquidity fee of 3%, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.
- ❖ The fee of 2% will be sent to a private wallet, which will be allocated to buyback tokens and charity. Buyback is used to save from massive dips in order to keep the token market price stable.

Tokenomics

15% tax fee when buying and selling

- 10% of every trade goes to holders pockets in BNB.
- 3% of every trade goes to the liquidity pool.
- 2% of every trade goes to buyback tokens and charity.

Roadmap

JULY, 2021	
✓	**Roadmap Completed**
✓	**Gitbook**
✓	**Token Launch**
✓	**Fairlaunch**
AUGUST, 2021	
✓	**Audit**
	List on CoinGecko
	List on CoinMarketCap
	Community Marketing Campaign
	NFTs Cat Box
	Partnership Announce
	Hire Influencer for promoting
September, 2021	
	Cats Breeding
	Cat's Farm
	Charity Donation for Stray Cats
	List on Exchange
	Certik Audit

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 BNB by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in taking part with upcoming NEKO ANGEL NFTs.
- Anyone who's interested in making financial transactions with any other party using NEKO ANGEL as the currency.

Core concept

The BNB reward system

10% of each transaction gets converted to BNBs, and is split amongst all holders. The rewards are sent to holders that have at least 10,000 NEKO ANGEL tokens, holders will be eligible to receive tokens and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The liquidity fee of 3%, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Good cause and keeping the market stable

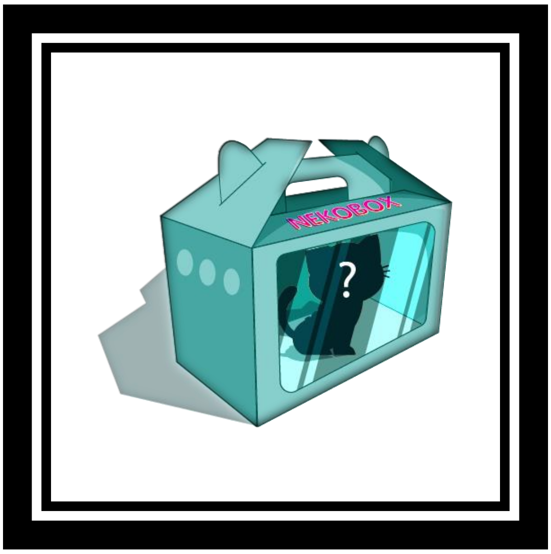
The **fee of 2%** tax on each transaction, which is stored inside a private wallet for buyback tokens or charity. The funds will be kept in the private wallet and will be used for charity or buyback as per the community preference.

Buyback is to save from massive dips in order to keep the token market price stable.

Anti-whale strategy

The NEKO ANGEL contract includes a function that stops all sales above 0.1% of the total supply. This will discourage (mini)-whales from dumping all their bags at once.

The upcoming features



To participate in NFTs;

- The $NekoAngel will be required in buying our Neko mystery box!,
- At this moment, there are 3 cutest little Cat models in Neko Mystery Box.
- Holders will be getting 1 random Cat per 1 Neko Mystery Box.
- The most important part!, after getting a Cat(s), users are free to raise Cat(s) with love, breed your Cat(s), and create a Cat Farm to earn more $BNB.

Potential to grow with score points

1.	Project efficiency	7/10
2.	Project uniqueness	7/10
3	Information quality	6/10
4	Service quality	7/10
5	System quality	6/10
6	Impact on the community	7/10
7	Impact on the business	7/10
8	Preparing for the future	7/10
Total Points		**6.75/10**

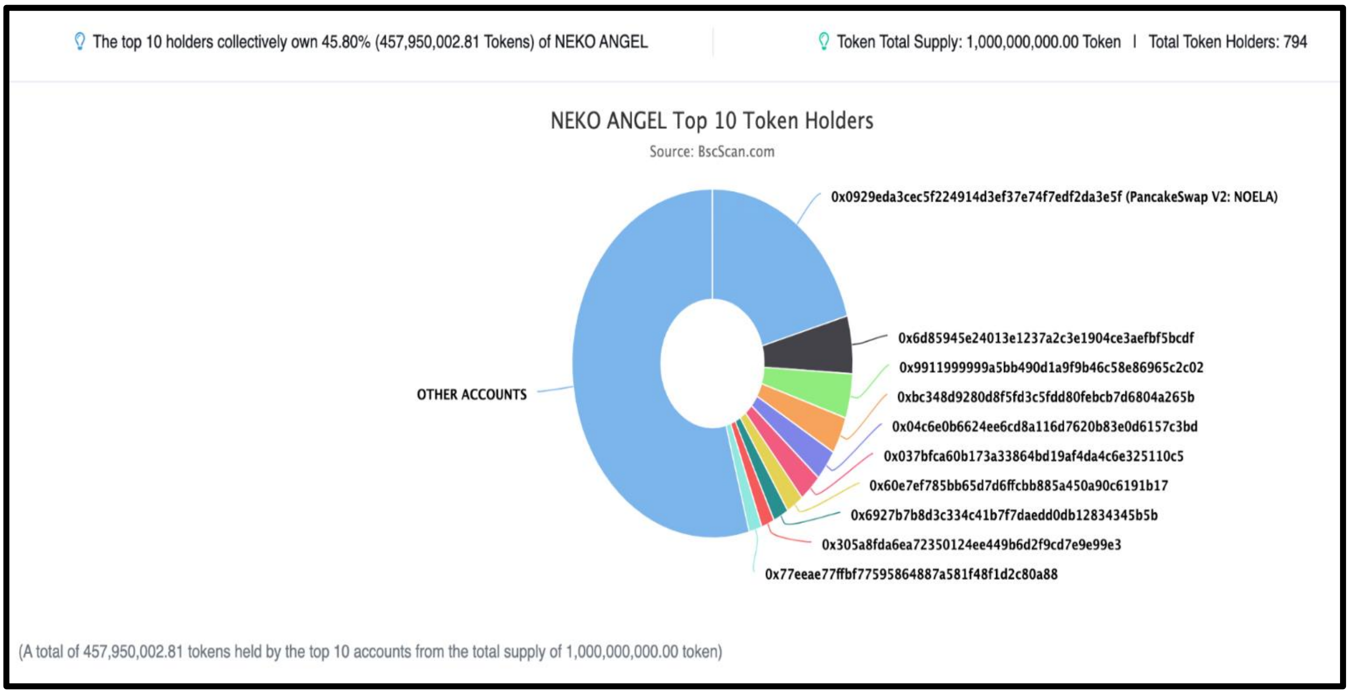
Contract details

Token contract details for 06th August 2021

Contract name	NEKO ANGEL Token
Contract address	0x37eBB7138122DF4954d14144b3a8E75279D89E3B
Token supply	1,000,000,000
Token ticker	NOELA
Decimals	18
Token holders	794
Transaction count	2,525
Top 100% holders dominance	87.28%
Dividend Tracker	0x6514d627a5fb5354ab4f5f6059f1afaad787c434
Liquidity wallet	0x6d85945e24013e1237a2c3e1904ce3aefbf5bcdf
Operation wallet	0x20f2005089d641b7b999a3c5dde25b10fc2b25c5
Contract deployer address	0x6d85945E24013E1237A2C3E1904CE3AEfBf5bcdf
Contract's current owner address	0x6d85945e24013e1237a2c3e1904ce3aefbf5bcdf

Top token holders

Top 10 Token Holders



The top 10 holders collectively own 45.80% (457,950,002.81 Tokens) of NEKO ANGEL

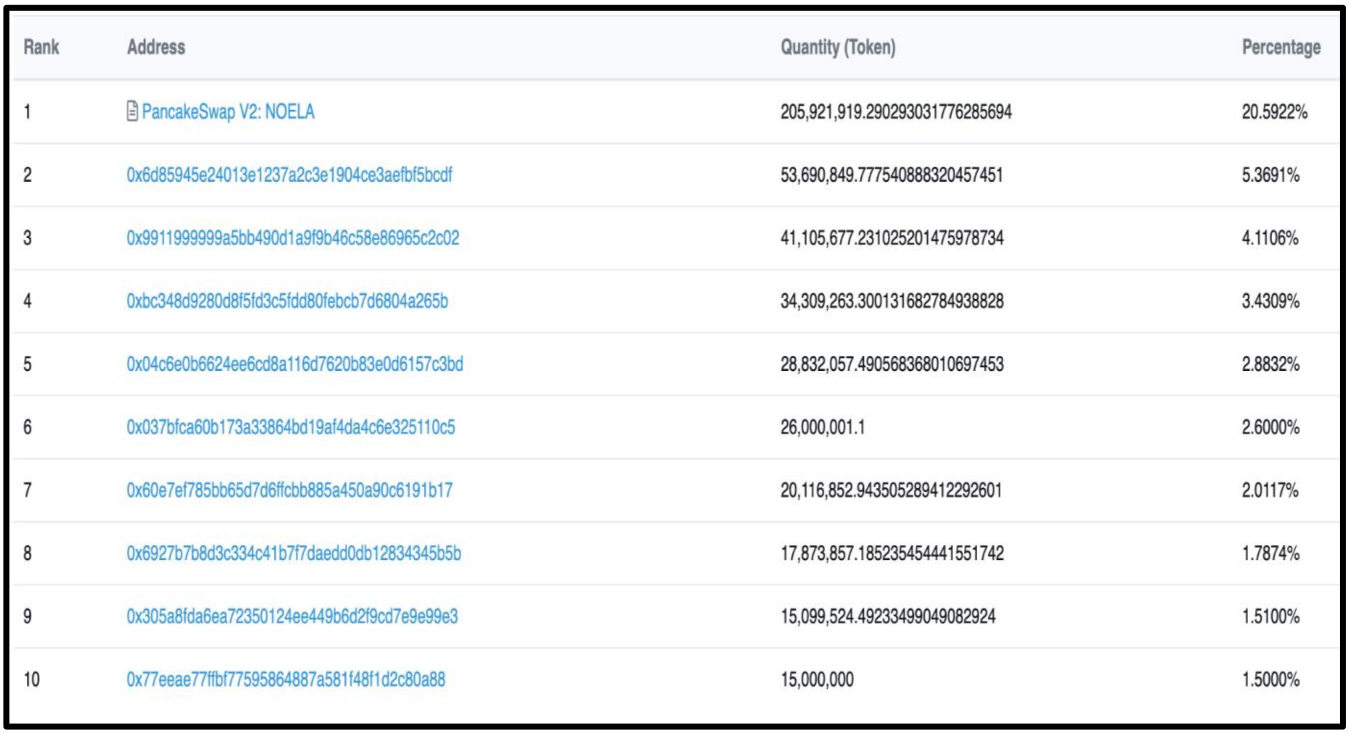
Token Total Supply: 1,000,000,000.00 Token | Total Token Holders: 794

NEKO ANGEL Top 10 Token Holders

Source: BscScan.com

0x0929eda3cec5f224914d3ef37e74f7edf2da3e5f (PancakeSwap V2: NOELA)
0x6d85945e24013e1237a2c3e1904ce3aefbf5bcdf
0x9911999999a5bb490d1a9f9b46c58e86965c2c02
0xbc348d9280d8f5fd3c5fdd80febcb7d6804a265b
0x04c6e0b6624ee6cd8a116d7620b83e0d6157c3bd
0x037bfca60b173a33864bd19af4da4c6e325110c5
0x60e7ef785bb65d7d6ffcbb885a450a90c6191b17
0x6927b7b8d3c334c41b7f7daedd0db12834345b5b
0x305a8fda6ea72350124ee449b6d2f9cd7e9e99e3
0x77eeae77ffbf77595864887a581f48f1d2c80a88
OTHER ACCOUNTS

(A total of 457,950,002.81 tokens held by the top 10 accounts from the total supply of 1,000,000,000.00 token)



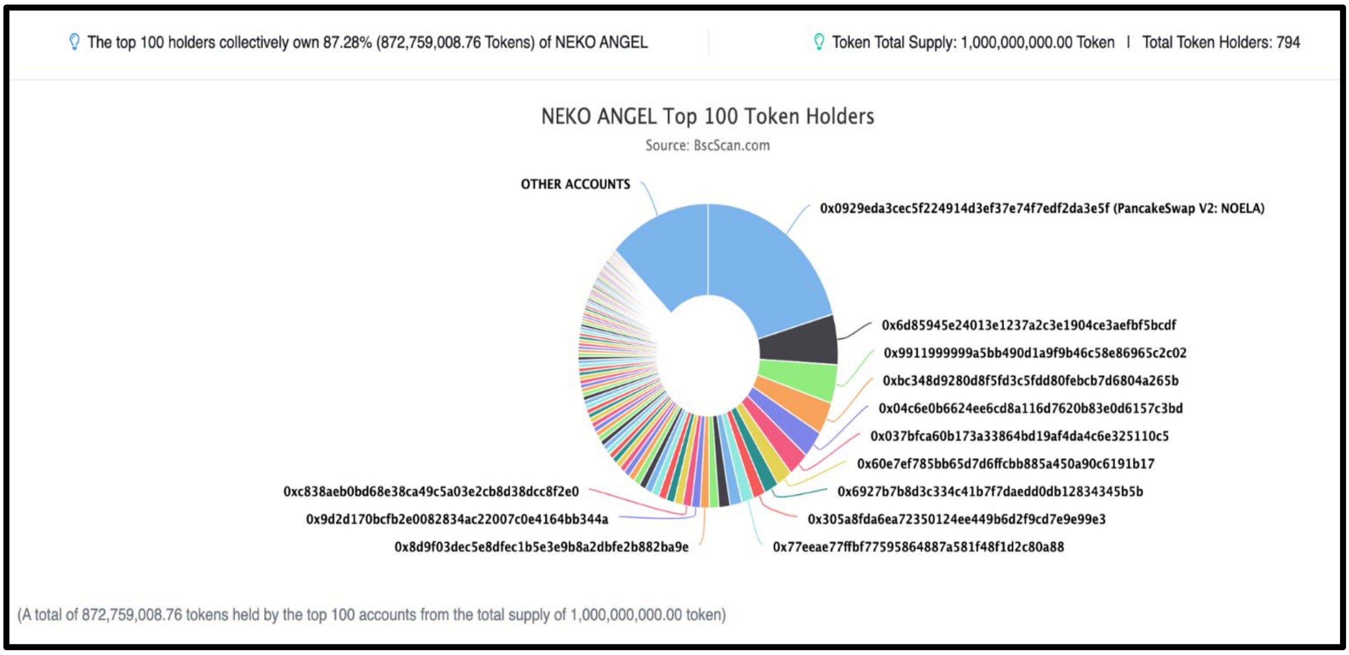
Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: NOELA	205,921,919.290293031776285694	20.5922%
2	0x6d85945e24013e1237a2c3e1904ce3aefbf5bcdf	53,690,849.777540888320457451	5.3691%
3	0x9911999999a5bb490d1a9f9b46c58e86965c2c02	41,105,677.231025201475978734	4.1106%
4	0xbc348d9280d8f5fd3c5fdd80febcb7d6804a265b	34,309,263.300131682784938828	3.4309%
5	0x04c6e0b6624ee6cd8a116d7620b83e0d6157c3bd	28,832,057.490568368010697453	2.8832%
6	0x037bfca60b173a33864bd19af4da4c6e325110c5	26,000,001.1	2.6000%
7	0x60e7ef785bb65d7d6ffcbb885a450a90c6191b17	20,116,852.943505289412292601	2.0117%
8	0x6927b7b8d3c334c41b7f7daedd0db12834345b5b	17,873,857.185235454441551742	1.7874%
9	0x305a8fda6ea72350124ee449b6d2f9cd7e9e99e3	15,099,524.49233499049082924	1.5100%
10	0x77eeae77ffbf77595864887a581f48f1d2c80a88	15,000,000	1.5000%

Token distribution

Tokens are distributed as follows:

❖	**Burnt**	**: 5%**
❖	**Contract**	**: 47%**
❖	**Dev Wallet**	**: 3%**

Top 100 Token Holders



The top 100 holders collectively own 87.28% (872,759,008.76 Tokens) of NEKO ANGEL

Token Total Supply: 1,000,000,000.00 Token | Total Token Holders: 794

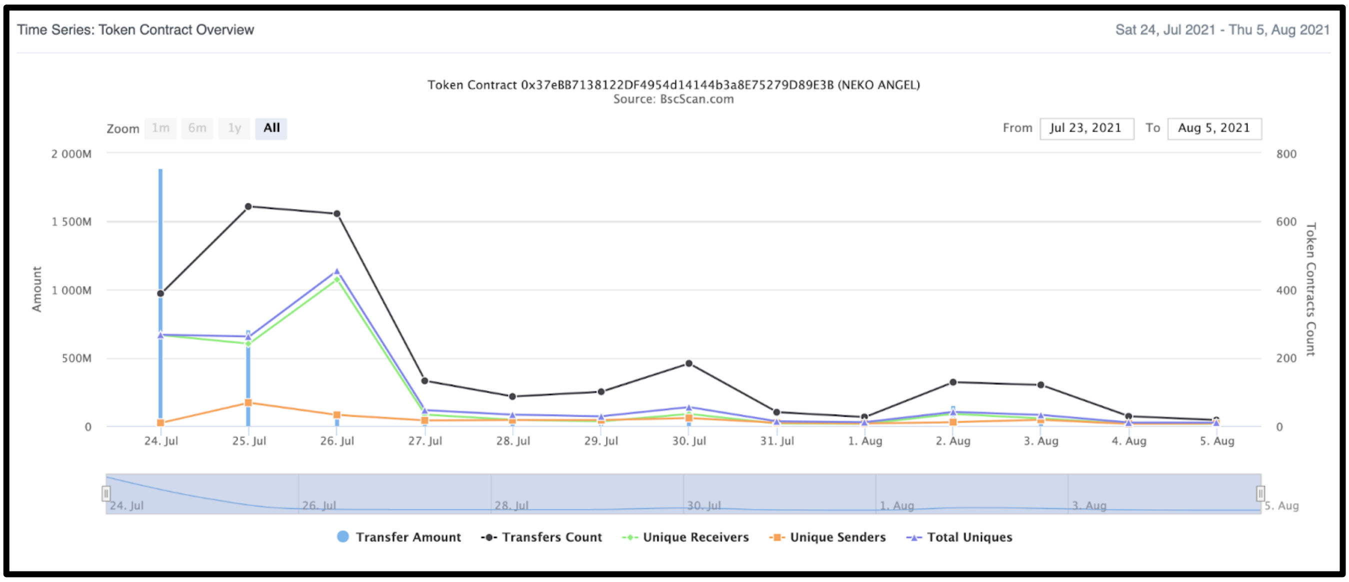
NEKO ANGEL Top 100 Token Holders

Source: BscScan.com

OTHER ACCOUNTS
0x0929eda3cec5f224914d3ef37e74f7edf2da3e5f (PancakeSwap V2: NOELA)
0x6d85945e24013e1237a2c3e1904ce3aefbf5bcdf
0x9911999999a5bb490d1a9f9b46c58e86965c2c02
0xbc348d9280d8f5fd3c5fdd80febcb7d6804a265b
0x04c6e0b6624ee6cd8a116d7620b83e0d6157c3bd
0x037bfca60b173a33864bd19af4da4c6e325110c5
0x60e7ef785bb65d7d6ffcbb885a450a90c6191b17
0x6927b7b8d3c334c41b7f7daedd0db12834345b5b
0x305a8fda6ea72350124ee449b6d2f9cd7e9e99e3
0x77eeae77ffbf77595864887a581f48f1d2c80a88
0xc838aeb0bd68e38ca49c5a03e2cb8d38dcc8f2e0
0x9d2d170bcfb2e0082834ac22007c0e4164bb344a
0x8d9f03dec5e8dfec1b5e3e9b8a2dbfe2b882ba9e

(A total of 872,759,008.76 tokens held by the top 100 accounts from the total supply of 1,000,000,000.00 token)

Contract interaction details



Time Series: Token Contract Overview

Sat 24, Jul 2021 - Thu 5, Aug 2021

Token Contract 0x37eBB7138122DF4954d14144b3a8E75279D89E3B (NEKO ANGEL)

Source: BscScan.com

Zoom 1m 6m 1y All

From Jul 23, 2021 To Aug 5, 2021

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

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**		
L	Function Name	Visibility	Mutability	Modifiers
NOELA	**Implementation**	**ERC20, Ownable**		
L		Public ❗	🛑	ERC20
L		External ❗	💵	NO❗
L	updateDividendTracker	Public ❗	🛑	onlyOwner
L	updateUniswapV2Router	Public ❗	🛑	onlyOwner
L	excludeFromFees	Public ❗	🛑	onlyOwner
L	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
L	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
L	_setAutomatedMarketMakerPair	Private 🔐	🛑	
L	updateLiquidityWallet	Public ❗	🛑	onlyOwner
L	updateOperationWallet	Public ❗	🛑	onlyOwner
L	updateGasForProcessing	Public ❗	🛑	onlyOwner
L	updateClaimWait	External ❗	🛑	onlyOwner
L	getClaimWait	External ❗		NO❗
L	getTotalDividendsDistributed	External ❗		NO❗

└	isExcludedFromFees	Public ❗		NO❗
└	withdrawableDividendOf	Public ❗		NO❗
└	dividendTokenBalanceOf	Public ❗		NO❗
└	getAccountDividendsInfo	External ❗		NO❗
└	getAccountDividendsInfoAtIndex	External ❗		NO❗
└	processDividendTracker	External ❗	🛑	NO❗
└	claim	External ❗	🛑	NO❗
└	getLastProcessedIndex	External ❗		NO❗
└	getNumberOfDividendTokenHolders	External ❗		NO❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	includeDividends	External ❗	🛑	onlyOwner
└	isPenaltyEnabled	Public ❗		NO❗
└	getTradingIsEnabled	Public ❗		NO❗
└	setCanTransferBeforeTradingIsEnabled	Public ❗	🛑	onlyOwner
└	setTradingEnabledTimestamp	Public ❗	🛑	onlyOwner
└	setPenaltyDisableBlock	Public ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	
└	swapAndLiquify	Private 🔐	🛑	
└	_swapTokensForEth	Private 🔐	🛑	

└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	swapAndSendOperation	Private 🔐	🛑	
└	swapAndSendDividends	Private 🔐	🛑	
NOELADividendTracker	**Implementation**	**DividendPayingToken, Ownable**		
└		Public ❗️	🛑	DividendPayingToken
└	_transfer	Internal 🔒	🛑	
└	withdrawDividend	Public ❗️	🛑	NO❗️
└	excludeFromDividends	External ❗️	🛑	onlyOwner
└	includeDividends	External ❗️	🛑	onlyOwner
└	updateClaimWait	External ❗️	🛑	onlyOwner
└	getLastProcessedIndex	External ❗️		NO❗️
└	getNumberOfTokenHolders	External ❗️		NO❗️
└	getAccount	Public ❗️		NO❗️
└	getAccountAtIndex	Public ❗️		NO❗️
└	canAutoClaim	Private 🔐		
└	setBalance	External ❗️	🛑	onlyOwner
└	process	Public ❗️	🛑	NO❗️
└	processAccount	Public ❗️	🛑	onlyOwner
DividendPayingToken	**Implementation**	**ERC20, DividendPayingTokenInterface,**		

		DividendPayingTokenOptionalInterface		
└		Public ❗️	🛑	ERC20
└		External ❗️	💵	NO❗️
└	distributeDividends	Public ❗️	💵	NO❗️
└	withdrawDividend	Public ❗️	🛑	NO❗️
└	_withdrawDividendOfUser	Internal 🔒	🛑	
└	dividendOf	Public ❗️		NO❗️
└	withdrawableDividendOf	Public ❗️		NO❗️
└	withdrawnDividendOf	Public ❗️		NO❗️
└	accumulativeDividendOf	Public ❗️		NO❗️
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_setBalance	Internal 🔒	🛑	
ERC20	**Implementation**	**Context, IERC20, IERC20Metadata**		
└		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❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
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 🔒		
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
SafeMathUint	**Library**			
└	toInt256Safe	Internal 🔒		
SafeMathInt	**Library**			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
DividendPayingTokenInterface	**Interface**			
└	dividendOf	External ❗		NO❗
└	distributeDividends	External ❗	💵	NO❗

└	withdrawDividend	External ❗	🛑	NO❗
DividendPayingTokenOptionalInterface	**Interface**			
└	withdrawableDividendOf	External ❗		NO❗
└	withdrawnDividendOf	External ❗		NO❗
└	accumulativeDividendOf	External ❗		NO❗
IterableMapping	**Library**			
└	get	Public ❗		NO❗
└	getIndexOfKey	Public ❗		NO❗
└	getKeyAtIndex	Public ❗		NO❗
└	size	Public ❗		NO❗
└	set	Public ❗	🛑	NO❗
└	remove	Public ❗	🛑	NO❗
Ownable	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
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❗
└	mint	External ❗	🛑	NO❗
└	burn	External ❗	🛑	NO❗
└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗

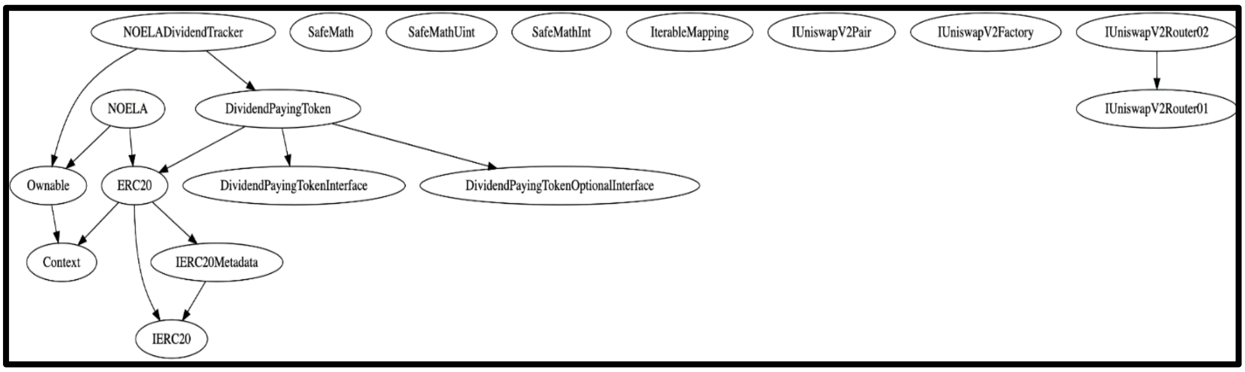
IUniswapV2Factory	**Interface**			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🛑	NO❗
└	setFeeTo	External ❗	🛑	NO❗
└	setFeeToSetter	External ❗	🛑	NO❗
IUniswapV2Router01	**Interface**			
└	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**	**IUniswapV2Router01**		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗

Legend

Symbol	Meaning
🛑	**Function can modify state**
💵	**Function is payable**

Inheritance Hierarchy



Security issue checking status

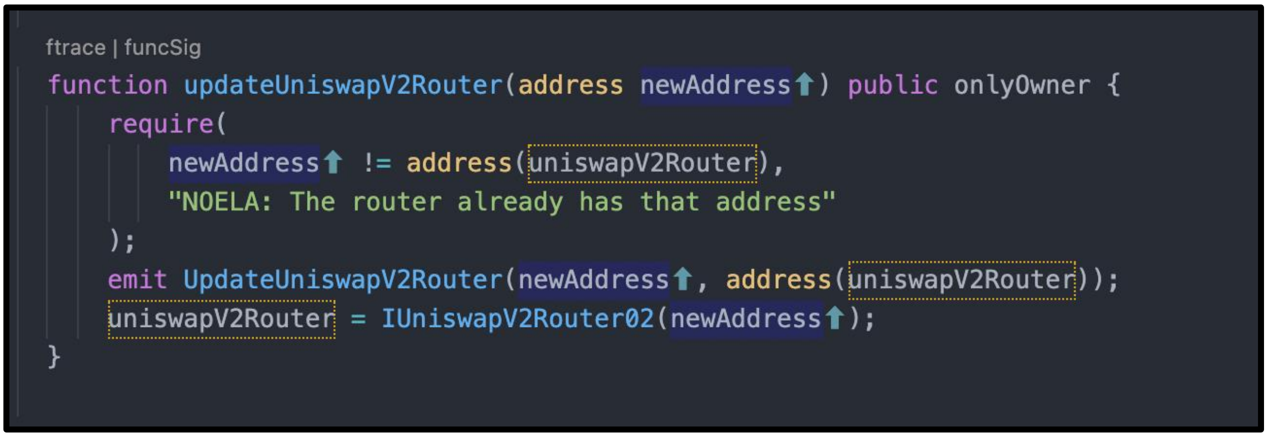
❖ **High severity issues**
No high severity issues found.

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

❖ **Low severity issues**
No medium severity issues found.

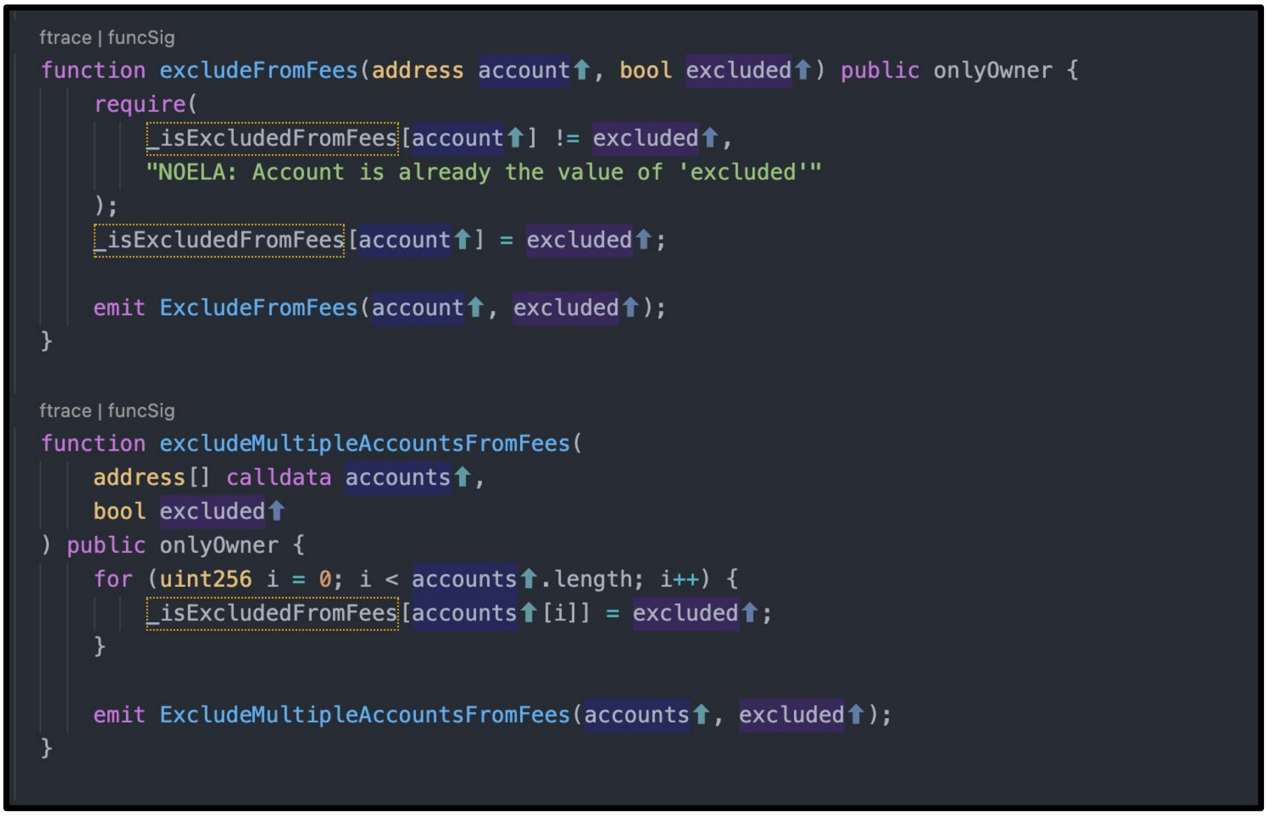
Owner privileges

- ❖ The owner can change the uniswap router address.



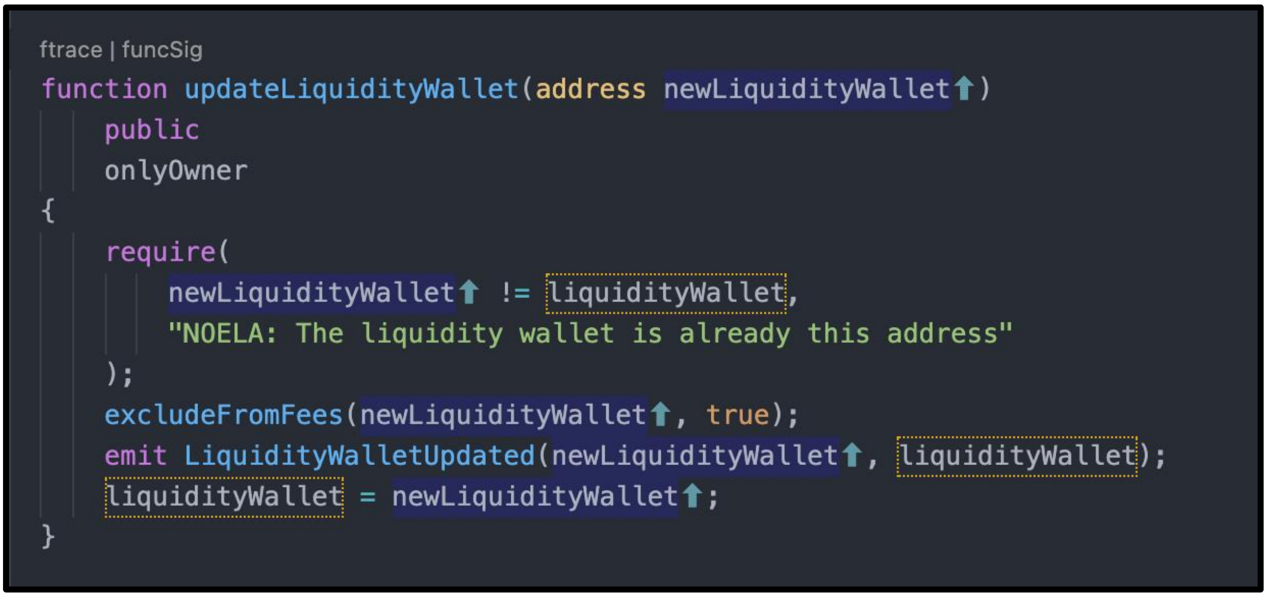
```
ftrace | funcSig
function updateUniswapV2Router(address newAddress) public onlyOwner {
    require(
        newAddress != address(uniswapV2Router),
        "NOELA: The router already has that address"
    );
    emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress);
}
```

- ❖ The owner can exclude accounts from fees.



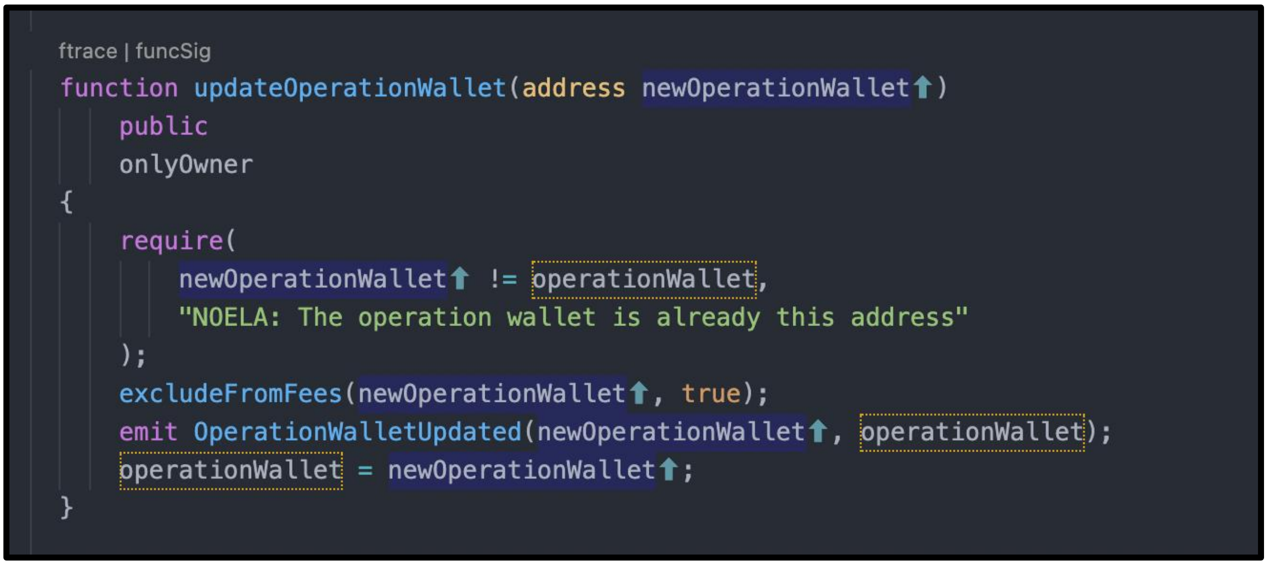
```
ftrace | funcSig
function excludeFromFees(address account, bool excluded) public onlyOwner {
    require(
        _isExcludedFromFees[account] != excluded,
        "NOELA: Account is already the value of 'excluded'"
    );
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}

ftrace | funcSig
function excludeMultipleAccountsFromFees(
    address[] calldata accounts,
    bool excluded
) public onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }

    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

- ❖ The owner can change the liquidity wallet.



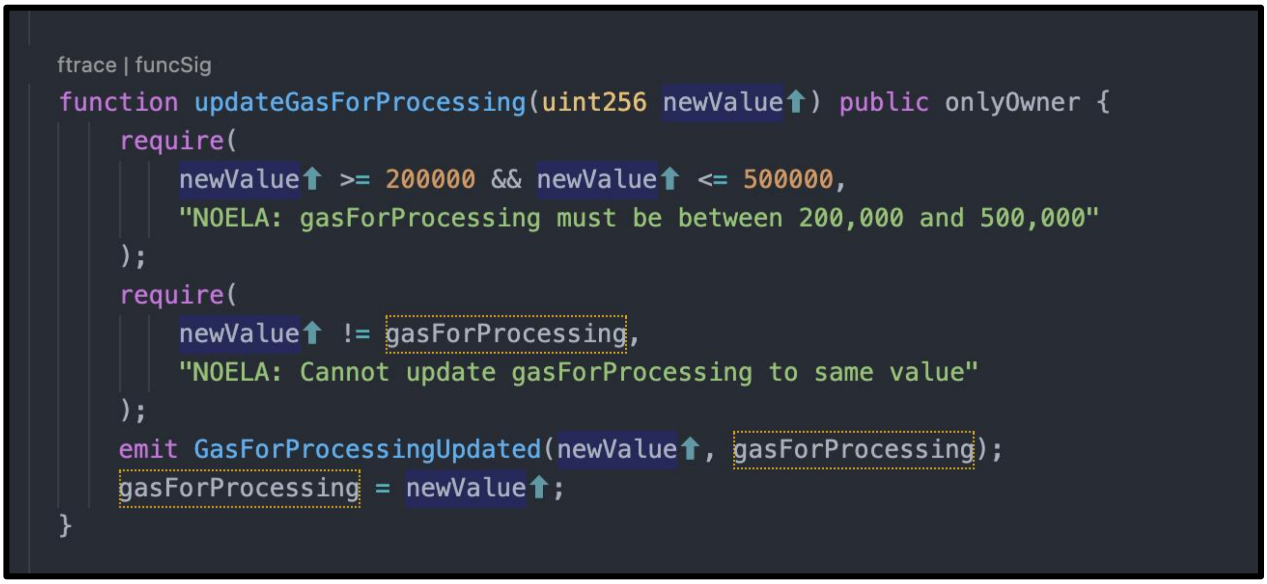
```
ftrace | funcSig
function updateLiquidityWallet(address newLiquidityWallet)
    public
    onlyOwner
{
    require(
        newLiquidityWallet != liquidityWallet,
        "NOELA: The liquidity wallet is already this address"
    );
    excludeFromFees(newLiquidityWallet, true);
    emit LiquidityWalletUpdated(newLiquidityWallet, liquidityWallet);
    liquidityWallet = newLiquidityWallet;
}
```

- ❖ The owner can change the operation wallet.



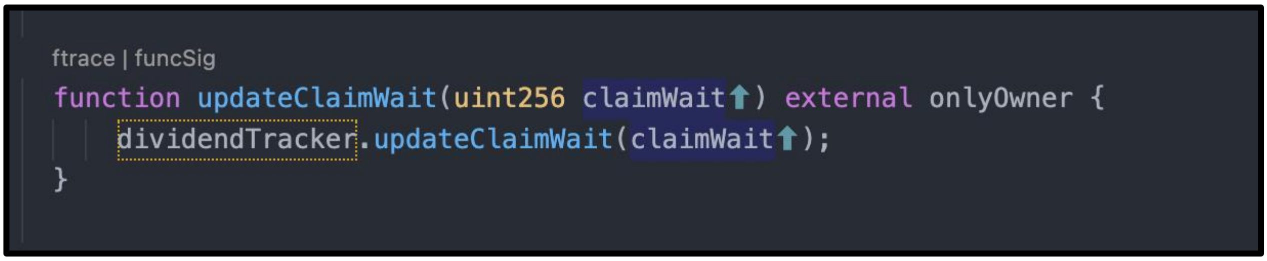
```
ftrace | funcSig
function updateOperationWallet(address newOperationWallet)
    public
    onlyOwner
{
    require(
        newOperationWallet != operationWallet,
        "NOELA: The operation wallet is already this address"
    );
    excludeFromFees(newOperationWallet, true);
    emit OperationWalletUpdated(newOperationWallet, operationWallet);
    operationWallet = newOperationWallet;
}
```

- The owner can change the gas fee from 200000 to 500000.



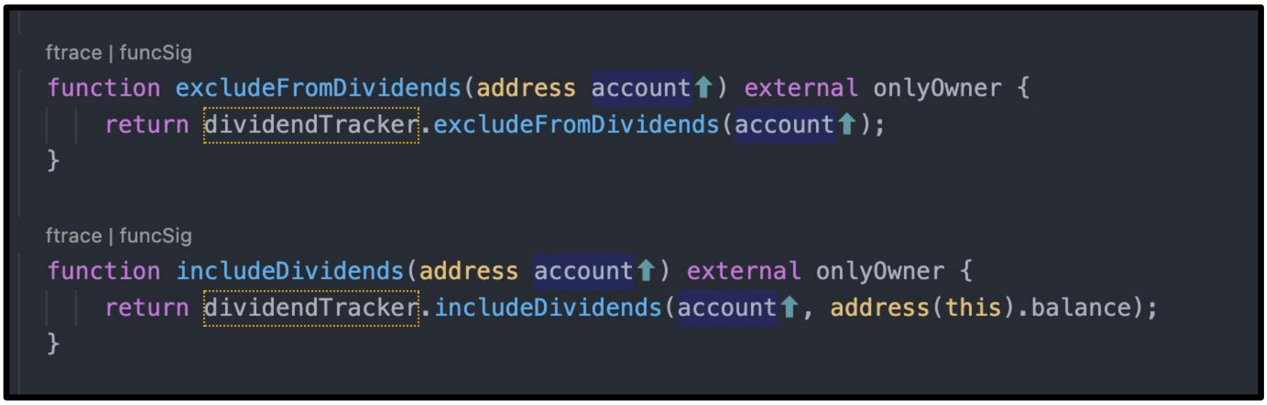
```
ftrace | funcSig
function updateGasForProcessing(uint256 newValue) public onlyOwner {
    require(
        newValue >= 200000 && newValue <= 500000,
        "NOELA: gasForProcessing must be between 200,000 and 500,000"
    );
    require(
        newValue != gasForProcessing,
        "NOELA: Cannot update gasForProcessing to same value"
    );
    emit GasForProcessingUpdated(newValue, gasForProcessing);
    gasForProcessing = newValue;
}
```

- The owner can change the claim wait.



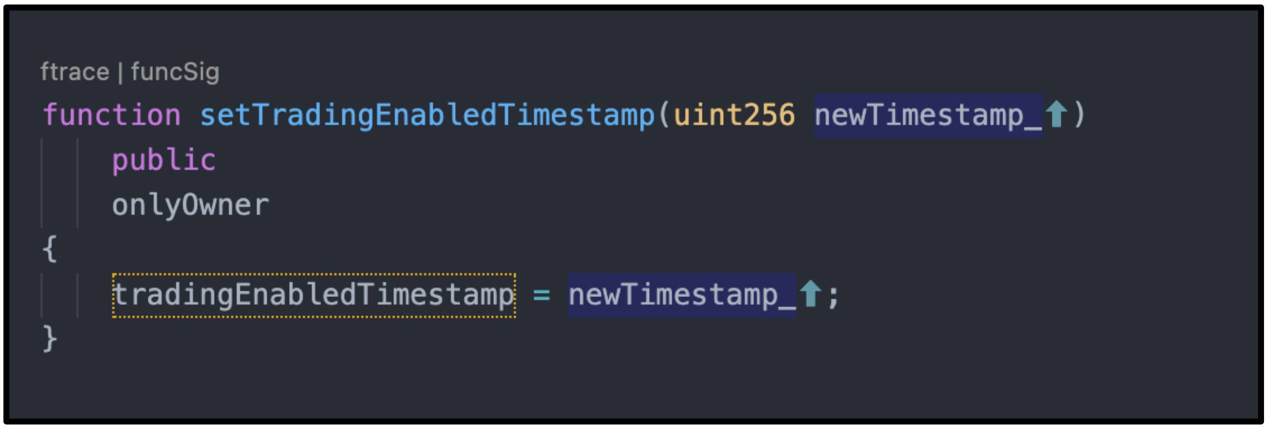
```
ftrace | funcSig
function updateClaimWait(uint256 claimWait) external onlyOwner {
    dividendTracker.updateClaimWait(claimWait);
}
```

- The owner can include/exclude wallets from dividends.



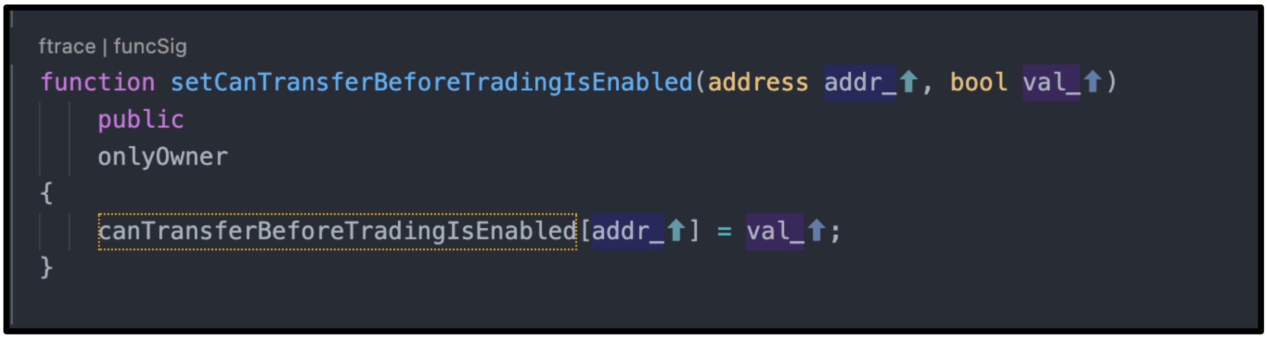
```
ftrace | funcSig
function excludeFromDividends(address account) external onlyOwner {
    return dividendTracker.excludeFromDividends(account);
}

ftrace | funcSig
function includeDividends(address account) external onlyOwner {
    return dividendTracker.includeDividends(account, address(this).balance);
}
```

- The owner can change trading enable time.



```
ftrace | funcSig
function setTradingEnabledTimestamp(uint256 newTimestamp_⬆)
    public
    onlyOwner
{
    tradingEnabledTimestamp = newTimestamp_⬆;
}
```

- The owner can whitelist wallets to send tokens when trading is disabled.



```
ftrace | funcSig
function setCanTransferBeforeTradingIsEnabled(address addr_⬆, bool val_⬆)
    public
    onlyOwner
{
    canTransferBeforeTradingIsEnabled[addr_⬆] = val_⬆;
}
```

Audit conclusion

While conducting the audit of the NEKO ANGEL smart contract, it was observed that there is nothing alarming with the code.