RugFreeCoins Audit

Metamoon Token

Smart Contract Security Audit

December 7, 2021

Contents

Audit details

Audited project

Metamoon Token

Contract Address

0x6378AAD8c6693F38beECB1C0dBc0194B99cf050b

Client contact

Metamoon Token Team

Blockchain

Binance smart chain

Project website

https://metamoontoken.org/

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 Metamoon Token to perform an audit of the smart contract.

https://bscscan.com/address/0x6378AAD8c6693F38beECB1C0dBc0194B99cf050b

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

Metamoon token is a deflationary token built on the Binance Smart Chain, NFT, and NFT game + Defi. As holders of Metamoon token receive 6% distribution in BNB, 2 percentage on all transactions auto liquidate and burn thereby making the value of Metamoon reduce drastically as more and more transaction is done. Metamoon is a longterm project with an enjoyable and entertaining P2E game. On metaverse, Metamoon will be the first blockchain game to give BNB back to its holders.

Features

- 6% of each transaction when buying and selling gets sent amongst all holders in BNB rewards. The holders will be eligible to receive BNB, every one hour, and rewards are proportional to how many tokens each individual holds.

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

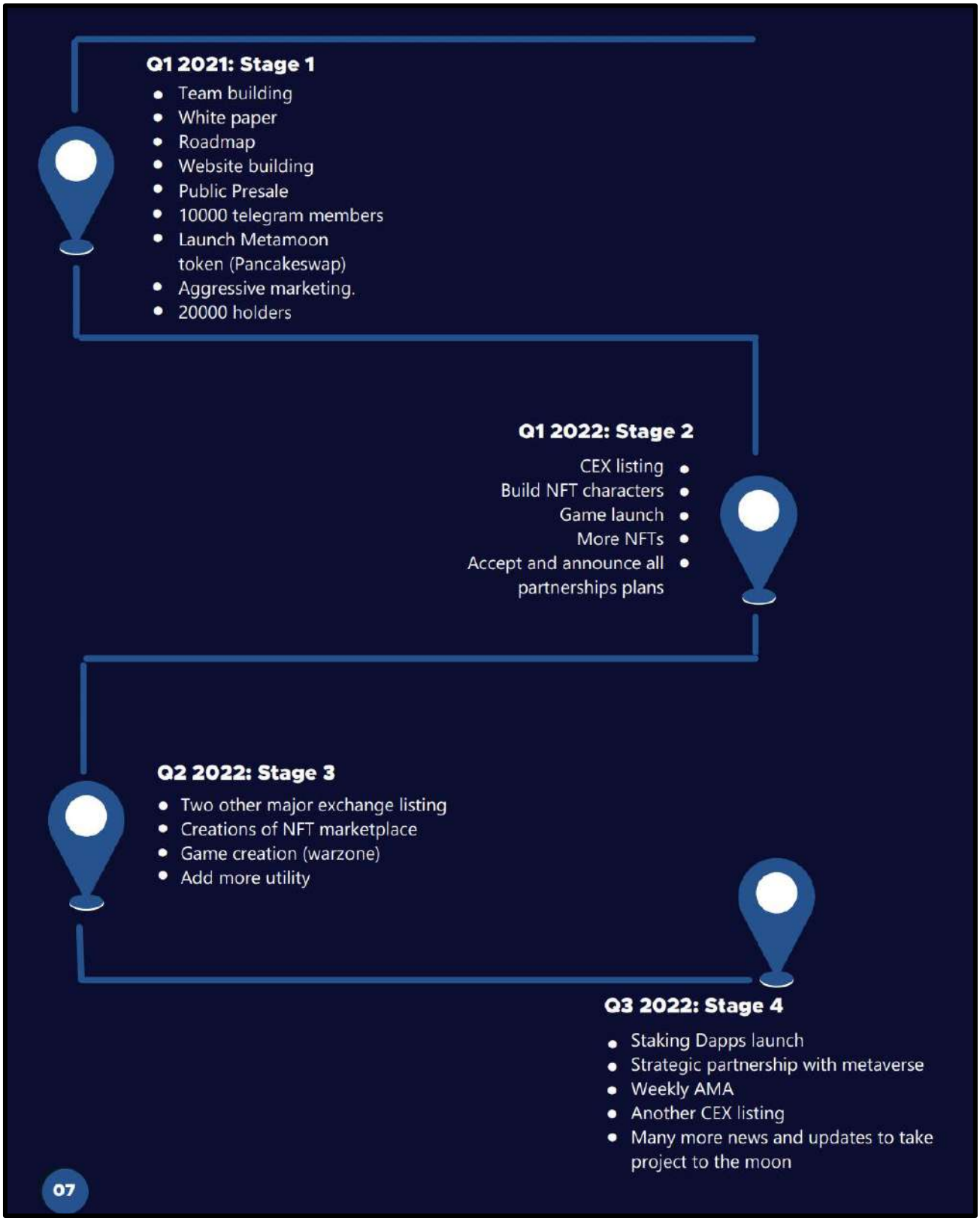
- The additional component included under the sustainability section is a **liquidity fee of 2% when buying and selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.

Tokenomics

10% fee when buying & selling

- 6% of trade goes to holders pockets in BNB.
- 2% of trade goes to the burn wallet.
- 2% of trade goes to development and marketing.

Roadmap


Q1 2021: Stage 1
Team building
White paper
Roadmap
Website building
Public Presale
10000 telegram members
Launch Metamoon token (Pancakeswap)
Aggressive marketing.
20000 holders
Q1 2022: Stage 2
CEX listing
Build NFT characters
Game launch
More NFTs
Accept and announce all partnerships plans
Q2 2022: Stage 3
Two other major exchange listing
Creations of NFT marketplace
Game creation (warzone)
Add more utility
Q3 2022: Stage 4
Staking Dapps launch
Strategic partnership with metaverse
Weekly AMA
Another CEX listing
Many more news and updates to take project to the moon
07

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 by holding tokens.
- Anyone who's interested in owning NFTs.
- Any casual or hardcore gamers out there play games and win rewards.
- Anyone who's interested in collecting NFTs or trading NFTs.
- Anyone who's interested in trading tokens.
- Anyone who's interested in staking Metamoon and get rewards.
- Anyone who's interested in taking part with the future plans of the Metamoon token.
- Anyone who's interested in making financial transactions with any other party using Metamoon or BNB as the currency.

Core concept

The BNB reward system

6% of each transaction when buying and selling gets sent amongst all holders in BNB rewards. The holders will be eligible to receive BNB, every one hour, and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The **sustainability fee of 2% when buying and selling is allocated to the marketing and development**. This is what allows Metamoon Token to promote the token and use funds to further the development of the platform. Tokens will be swapped into BNB and will be sent to a marketing wallet. This way, Metamoon Token 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 liquidity fee of 2% when buying and selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity

Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	9/10
3	Information quality	8/10
4	Service quality	8/10
5	System quality	8/10
6	Impact on the community	9/10
7	Impact on the business	9/10
8	Preparing for the future	9/10
Total Points		**8.625/10**

Contract details

Token contract details for 07th December 2021

Contract name	Metamoon
Contract address	0x6378AAD8c6693F38beECB1C0dBc0194B99cf050b
Token supply	200,000,000
Token ticker	MMT
Decimals	18
Token holders	1
Transaction count	3
Marketing wallet address	0x95e65e95645937695028143e56491d5484d5e9e7
Dividend tracker	0x97dee8e8f8d084cab213cab6499bc3ff36da2512
Contract deployer address	0x3C998FA92f7cEFE27752f78c0e49B092EaD8D81c
Contract's current owner address	0xa41204bdd42eb910fb02d0725027adad589dda59

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	low issue
		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
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 🔒		
DividendPayingToken Interface	**Interface**			
└	dividendOf	External ❗		NO❗
└	distributeDividends	External ❗	💸	NO❗
└	withdrawDividend	External ❗	🛑	NO❗
DividendPayingToken OptionalInterface	**Interface**			
└	withdrawableDividendOf	External ❗		NO❗
└	withdrawnDividendOf	External ❗		NO❗
└	accumulativeDividendOf	External ❗		NO❗
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 🔒	🛑	
IterableMapping	**Library**			
└	get	Public ❗		NO❗
└	getIndexOfKey	Public ❗		NO❗
└	getKeyAtIndex	Public ❗		NO❗
└	size	Public ❗		NO❗

└	set	Public ❗	🔴	NO❗
└	remove	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❗
└	mint	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❗

└	swapExactTokens ForETH	External ❗	🛑	NO❗
└	swapETHForExact Tokens	External ❗	💸	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗
IUniswapV2Router02	**Interface**	**IUniswapV2Router01**		
└	removeLiquidityETH SupportingFeeO nTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETH WithPermitSupp ortingFeeOnTransf erTokens	External ❗	🛑	NO❗
└	swapExactTokens ForTokensSupport ingFeeOnTransfer Tokens	External ❗	🛑	NO❗
└	swapExactETHFor TokensSupporting FeeOnTransferTo kens	External ❗	💸	NO❗
└	swapExactTokens ForETHSupporting FeeOnTransferTo kens	External ❗	🛑	NO❗
Ownable	**Implementation**	**Context**		

└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
Metamoon	**Implementation**	**ERC20, Ownable**		
└		Public ❗	🛑	ERC20
└		External ❗	💸	NO❗
└	updateDividendTracker	Public ❗	🛑	onlyOwner
└	updateUniswapV2Router	Public ❗	🛑	onlyOwner
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setBNBRewardsFee	External ❗	🛑	onlyOwner
└	setLiquiditFee	External ❗	🛑	onlyOwner
└	setMarketingFee	External ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	blacklistAddress	External ❗	🛑	onlyOwner

└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	updateGasForProcessing	Public ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getClaimWait	External ❗		NO❗
└	getTotalDividendsDistributed	External ❗		NO❗
└	isExcludedFromFees	Public ❗		NO❗
└	withdrawableDividendOf	Public ❗		NO❗
└	dividendTokenBalanceOf	Public ❗		NO❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	setBuyFees	Public ❗	🛑	onlyOwner
└	setSellFees	Public ❗	🛑	onlyOwner
└	setFeeOnBuy	Private 🔐	🛑	
└	setFeeOnSell	Private 🔐	🛑	
└	getAccountDividendsInfo	External ❗		NO❗
└	getAccountDividendsInfoAtIndex	External ❗		NO❗
└	processDividendTracker	External ❗	🛑	NO❗
└	claim	External ❗	🛑	NO❗
└	getLastProcessedIndex	External ❗		NO❗

└	getNumberOfDividendTokenHolders	External ❗		NO❗
└	setExcludedFromAntiWhale	Public ❗	🛑	onlyOwner
└	isExcludedFromAntiWhale	Public ❗		NO❗
└	_transfer	Internal 🔒	🛑	antiWhale
└	maxTransferAmount	Public ❗		NO❗
└	setMaxTransferAmountRate	Public ❗	🛑	onlyOwner
└	swapAndSendToFee	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	swapAndSendDividends	Private 🔐	🛑	
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 🔒	🛑	
MMTDividendTracker	**Implementation**	**Ownable, DividendPayingToken**		
└		Public ❗	🛑	DividendPayingToken
└	_transfer	Internal 🔒	🛑	
└	withdrawDividend	Public ❗	🛑	NO❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗		NO❗
└	getNumberOfTokenHolders	External ❗		NO❗

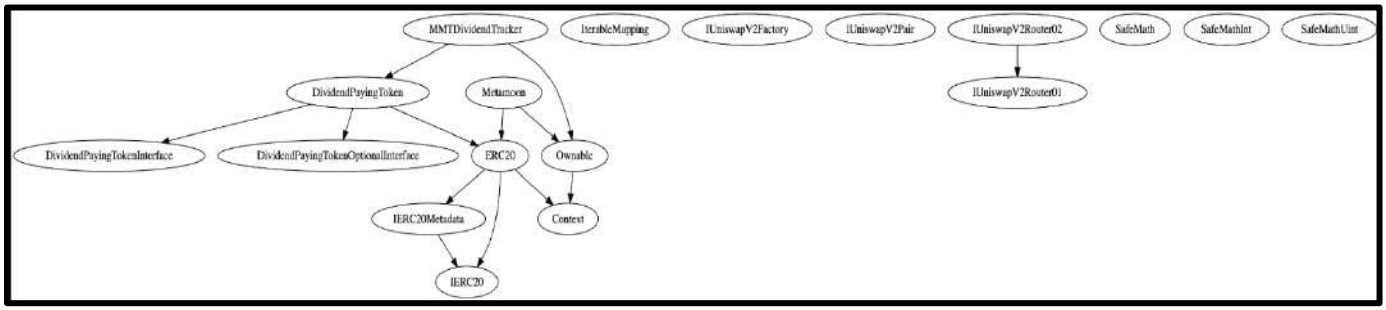
└	getAccount	Public ❗		NO❗
└	getAccountAtIndex	Public ❗		NO❗
└	canAutoClaim	Private 🔐		
└	setBalance	External ❗	🛑	onlyOwner
└	process	Public ❗	🛑	NO❗
└	processAccount	Public ❗	🛑	onlyOwner
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
SafeMathInt	**Library**			
└	mul	Internal 🔒		

└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
SafeMathUint	**Library**			
└	toInt256Safe	Internal 🔒		

Legend

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

Inheritance Hierarchy



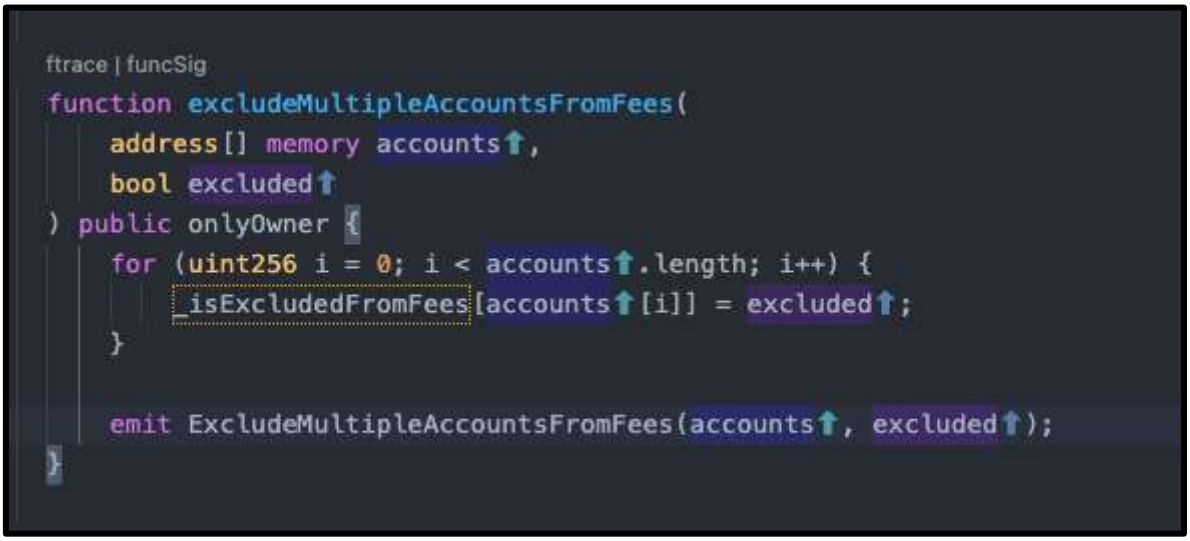
Security issue checking status

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

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

❖ **Low severity issues**

- In the excludeMultipleAccountsFromFees 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.

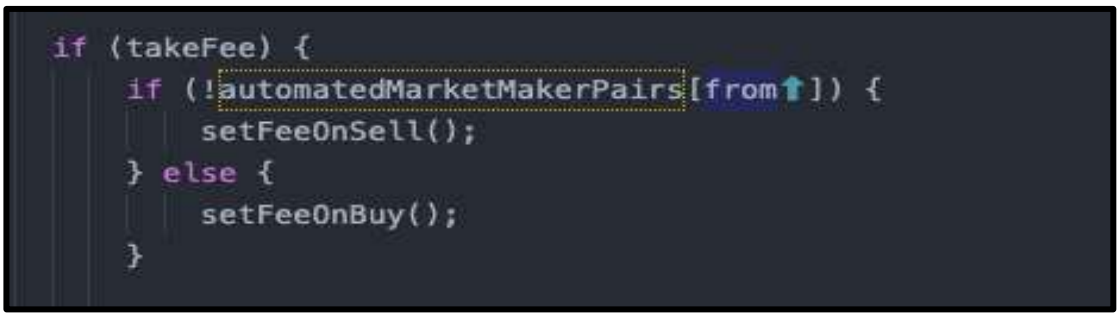


```
ftrace | funcSig
function excludeMultipleAccountsFromFees(
    address[] memory accounts⬆,
    bool excluded⬆
) public onlyOwner {
    for (uint256 i = 0; i < accounts⬆.length; i++) {
        _isExcludedFromFees[accounts⬆[i]] = excluded⬆;
    }

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

- **Wrong Logic when applying the selling fee**.

  The contract is checking whether the transaction is a “buy,” if it’s not a buy, it’s considered as a sell. Hence, the selling fee is applied when transferring from wallet to wallet.



```
if (takeFee) {
    if (!automatedMarketMakerPairs[from⬆]) {
        setFeeOnSell();
    } else {
        setFeeOnBuy();
    }
```

Owner privileges

- The owner can update the dividend tracker.

```
ftrace | funcSig
function updateDividendTracker(address newAddress⬆) public onlyOwner {
    require(
        newAddress⬆ != address(dividendTracker),
        "MMT: The dividend tracker already has that address"
    );

    MMTDividendTracker newDividendTracker = MMTDividendTracker(
        payable(newAddress⬆)
    );

    require(
        newDividendTracker.owner() == address(this),
        "MMT: The new dividend tracker must be owned by the MMT token contract"
    );

    newDividendTracker.excludeFromDividends(address(newDividendTracker));
    newDividendTracker.excludeFromDividends(address(this));
    newDividendTracker.excludeFromDividends(owner());
    newDividendTracker.excludeFromDividends(address(uniswapV2Router));

    emit UpdateDividendTracker(newAddress⬆, address(dividendTracker));

    dividendTracker = newDividendTracker;
}
```

- The owner can update the router address.

```
ftrace | funcSig
function updateUniswapV2Router(address newAddress⬆) public onlyOwner {
    require(
        newAddress⬆ != address(uniswapV2Router),
        "MMT: The router already has that address"
    );
    emit UpdateUniswapV2Router(newAddress⬆, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress⬆);
    address _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory())
        .createPair(address(this), uniswapV2Router.WETH());
    uniswapV2Pair = _uniswapV2Pair;
}
```

❖ The owner can exclude wallets from fees.

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

    emit ExcludeFromFees(account⬆, excluded⬆);
}
```

❖ The owner can change the marketing wallet address.

```
ftrace | funcSig
function setMarketingWallet(address payable wallet⬆) external onlyOwner {
    _marketingWalletAddress = wallet⬆;
}
```

❖ The owner can change all fees.

```
ftrace | funcSig
function setBNBRewardsFee(uint256 value⬆) external onlyOwner {
    require(value⬆ <= 10, "max 10%");
    BNBRewardsFee = value⬆;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee);
}

ftrace | funcSig
function setLiquiditFee(uint256 value⬆) external onlyOwner {
    require(value⬆ <= 5, "max 5%");
    liquidityFee = value⬆;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee);
}

ftrace | funcSig
function setMarketingFee(uint256 value⬆) external onlyOwner {
    require(value⬆ <= 5, "max 5%");
    marketingFee = value⬆;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee);
}
```

❖ The owner can blacklist wallets.

```
ftrace | funcSig
function blacklistAddress(address account⬆, bool value⬆) external onlyOwner {
    _isBlacklisted[account⬆] = value⬆;
}
```

❖ The owner can update gas fees and claim time in reward.

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

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

❖ The owner can exclude wallets from dividends.

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

❖ The owner can change all buy and sell fees.

```
ftrace | funcSig
function setBuyFees(
    uint256 reflectFee⬆,
    uint256 marketFee⬆,
    uint256 liquidFee⬆
) public onlyOwner {
    require(reflectFee⬆ <= 10, "max 10% reflect");
    require(marketFee⬆ <= 5, "max 5% market");
    require(liquidFee⬆ <= 5, "max 5% liquid");
    _buyFees._reflectFee = reflectFee⬆;
    _buyFees._marketFee = marketFee⬆;
    _buyFees._liquidFee = liquidFee⬆;
}

ftrace | funcSig
function setSellFees(
    uint256 reflectFee⬆,
    uint256 marketFee⬆,
    uint256 liquidFee⬆
) public onlyOwner {
    require(reflectFee⬆ <= 10, "max 10% reflect");
    require(marketFee⬆ <= 5, "max 5% market");
    require(liquidFee⬆ <= 5, "max 5% liquid");
    _sellFees._reflectFee = reflectFee⬆;
    _sellFees._marketFee = marketFee⬆;
    _sellFees._liquidFee = liquidFee⬆;
}
```

❖ The owner can change the max transaction amount.

```
ftrace | funcSig
function setMaxTransferAmountRate(uint256 _val⬆) public onlyOwner {
    require(_val⬆ <= 50000, "max 5%");
    maxTransferAmountRate = _val⬆;
}
```

Audit conclusion

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