



# **RugFreeCoins Audit**



## **Meta snails Token**

### **Smart Contract Security Audit**

**February 20, 2022**



# Contents

Audit details	1
Disclaimer	2
Background	3
About the project	4
Target market and the concept	9
Potential to grow with score points	15
Total Points	15
Contract details	16
Contract code function details	18
Contract description table	20
Security issue checking status	28
Audit conclusion	34

# Audit details



**Audited project**  
Meta snails Token



**Contract Address**  
0xb112658b35CAa05e65d6a4958b99738A14dFa7E4



**Client contact**  
Meta snailsTeam



**Blockchain**  
Binance smart chain



**Project website**  
<https://metasnails.io/>

# 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 Meta Snails Team to perform an audit of the smart contract.

**<https://bscscan.com/token/0xb112658b35CAa05e65d6a4958b99738A14dFa7E4>**

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

Meta Snails 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 of staking, P2E games, and NFTs. Each transaction, purchase incurs 0% fee, and sale incurs a 11% fee.

## Features

- The **sustainability fee of 1% when buying for game pool and 9% when selling for marketing and manual buyback** is what allows Meta Snails to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to marketing wallet. This way, Meta Snails 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 1% from selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

## Tokenomics

### 1% fee when buying

- 1% of trade goes to the marketing wallet in BNB allocated for game pool.

### 10% fee when selling

- 9% of trade goes to the marketing wallet in BNB allocated for marketing and manual buyback.
- 1% of trade goes to liquidity pool.

## Roadmap

### Q1

- ✓ Website launch
- ✓ contract security audit
- ✓ Team KYC
- ✓ Metasnails Token Private sale and NFT presale
- 🕒 NFT market Place
- 🕒 Launch MS coin LP lock 1 year

## Q2

- 🕒 Listing CoinMarketcap & Coingecko.
- 🕒 Major Exchanges incoming top 10. including gate.io ...
- 🕒 BillBoards & Ads in the whole Europe/US.
- 🕒 Beta Testers Program start and game testing mode.
- 🕒 MetaSnails Swag Merchandise Store.
- 🕒 Influencer Awareness Partnerships
- 🕒 Launch MetaSnails game on web browser and Windows PC.



## Q3

- 🕒 Token Staking
- 🕒 NFT stacking
- 🕒 Users NFT marketplace
- 🕒 2nd and 3rd security audit through Certik & Solidproof
- 🕒 24/7 DexTools & CMC & Poocoin & Crypto.com Trending.
- 🕒 Expand MetaSnails Game Modes + big Updates behind it.

## Q4

- 🕒 launch Meta snails game on android and IOS
- 🕒 Toldest holder contest ( Giveaway Tesla Model . )
- 🕒 NFT Contest
- 🕒 ETH Contract Bridging
- 🕒 MetaSnails Multichain wallet V1.0
- 🕒 Listing on tier 1 CEXs ( binance or coinbase or Kucoin )

Roadmap Update

# Target market and the concept

## Target market

- Anyone who's interested in the Crypto space with long-term investment plans.
- Anyone who's interested in trading tokens.
- Anyone who's interested in collecting NFTs or trading NFTs.
- Anyone who's interested in staking tokens and earn rewards.
- Anyone who's interested in taking part with the future plans of the Meta Snails token.
- Anyone who's interested in taking part with Meta Snails play to earn game and win rewards.
- Anyone who's interested in making financial transactions with any other party using Meta Snails as the currency

## Core concept

### Sustainable mechanism

The **sustainability fee of 1% when buying for game pool and 9% when selling for marketing and manual buyback** is what allows Meta Snails to promote the token and use funds to further the development of the platform. Tokens will be swapped into BNB and getting sent to the marketing wallet. This way, Meta Snailstoken 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 1% when selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

## Play to earn game

metasnails game allows investors to get passive income through different gaming modes that feeds each gamer/gamers needs, those modes are as follows :

**Free to PLAY mode :** this mode is manual and free to play with the lowest profitability ratio , player can only play with 1 free character

**auto Farming mode :** an auto-farm / play mode , no presence needed , activate manually each 24hrs with medium profitability

**manual mode :** an adventure mode that allows players to play manually with easy controls on mobile and windows , this mode has high profitability with the opportunity to win new items and collectibles while playing

playing metasnails nft game will allow players to earn ms tokens and gems that can be converted into several stables and coins through the game app, such as BNB , BUSD , USDT , BTC and ETH

## Game characters

### comon category :



Power : 1  
Stamina : 4  
Speed : 3  
Shield : 2



Power : 6  
Stamina : 5  
Speed : 3  
Shield : 2



Power : 7  
Stamina : 3  
Speed : 3  
Shield : 1



Power : 4  
Stamina : 2  
Speed : 4  
Shield : 5



Power : 4  
Stamina : 3  
Speed : 3  
Shield : 2



Power : 5  
Stamina : 2  
Speed : 4  
Shield : 3



## Rare category Y : ( animated )



Power : 10  
Stamina : 7  
Speed : 8  
Shield : 7



Power : 10  
Stamina : 7  
Speed : 8  
Shield : 7



Power : 15  
Stamina : 10  
Speed : 7  
Shield : 6

## Legend category Y : ( animated )



Power : 25  
Stamina : 19  
Speed : 17  
Shield : 19



Power : 20  
Stamina : 20  
Speed : 20  
Shield : 20



Power : 19  
Stamina : 20  
Speed : 25  
Shield : 18

## NFT marketplace

metasnails NFTs can be minted from our minting DAPP, each NFT has utility on our game including certain advantages and differences in power, stamina etc. Those characters can be minted before launch with BNB coin, after launch the minting will be available only with metasnails token.

Characters will be used in our game, each character has a category and a farming ratio that can be calculated with our profit calculator.

### Packs and single NFT :

Players can mint a pack of random NFT characters with certain market price.

or mint a character of a choice with a different market price.

100K NFTs will be the max supply ever.

## marketPLACE :

ALL NFTS MINTED THROUGH MS NFT DAPP CAN BE SOLDE IN MS FUTURE marketPLACE

ONLY 100K NFTS CAN BE EVER MINTED ON MS DAPP , AFTER THE contract runs out , THE NEW COMMERS CAN ONLY BUY THOSE NFTS FROM OTHER REAL INVESTORS ON OUR NFT marketPLACE , ALONG WITH OTHER UPCOMING UPGRADING ITEMS AND COSMETICS .

## NFT STAKING :

GET FREE NFTS (SHAILS) BY STACKING YOUR MS COIN TOKENS !!

NFT STACKING WILL ALLOW USERS TO STAKE CERTAIN AMOUNTS OF MS COIN TOKENS FOR NFT CHARACTERS

DETAILS AND NFT STACKING LAUNCHES ON Q3

# Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	9/10
3	Information quality	9/10
4	Service quality	9/10
5	System quality	9/10
6	Impact on the community	8/10
7	Impact on the business	9/10
8	Preparing for the future	9/10
Total Points		<b>8.875/10</b>

# Contract details

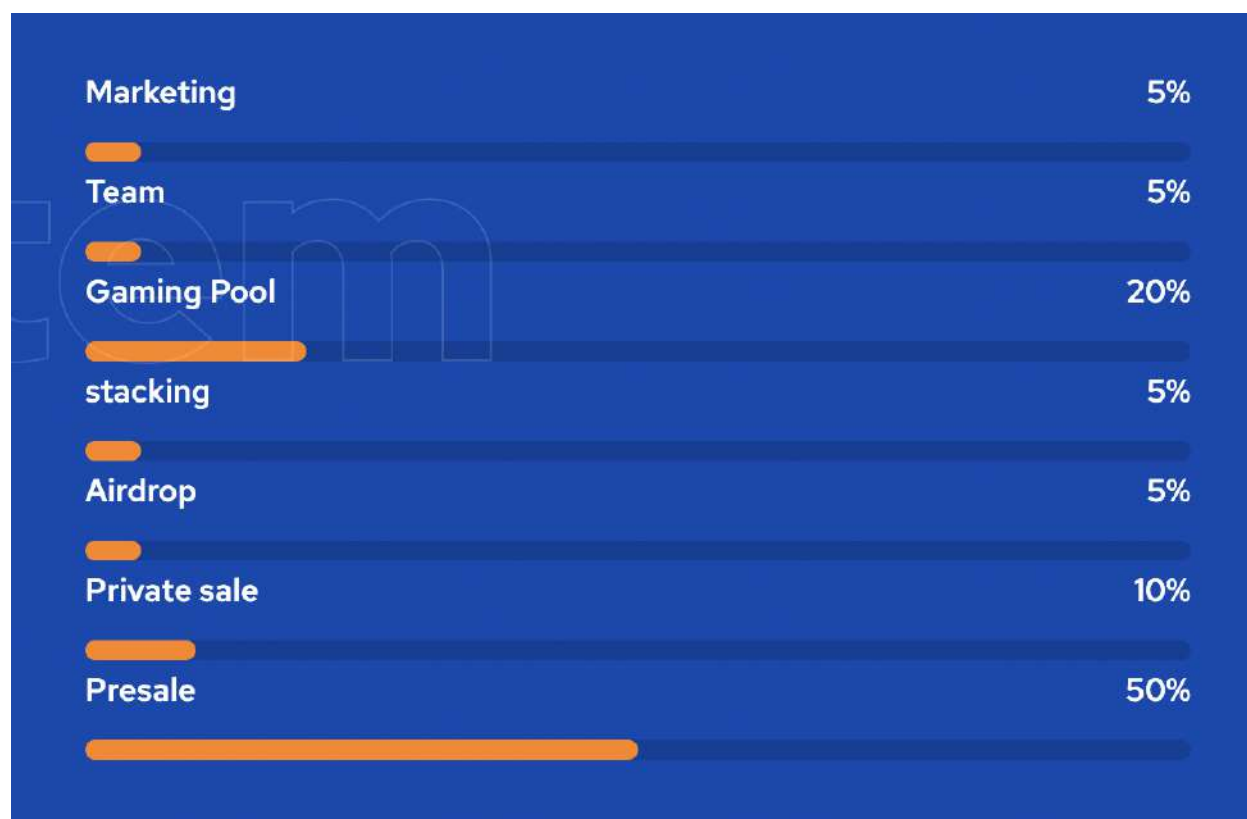
## Token contract details for 20<sup>th</sup> February 2022

Contract name	Meta Snails
Contract address	0xb112658b35CAa05e65d6a4958b99738A14dFa7E4
Token supply	120,000,000
Token ticker	MS
Decimals	9
Token holders	1
Transaction count	2
Marketing wallet	0xb0333d5cc08b6fe16d7ef4859856a6c4b499cdb4
Contract deployer address	0xC1660DB862cf3FD63AaE835BB7fF299a7bdBD639
Contract's current owner address	0x799ed71af53022c43955d6bda505a8789accb6a2



## Token distribution for

Tokens are distributed as follows:





















# 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
<b>IERC20</b>	<b>Interface</b>			
L	totalSupply	External 		NO 
L	balanceOf	External 		NO 
L	transfer	External 		NO 
L	allowance	External 		NO 
L	approve	External 		NO 
L	transferFrom	External 		NO 
<b>SafeMath</b>	<b>Library</b>			
L	add	Internal 		
L	sub	Internal 		
L	sub	Internal 		




























L	mul	Internal 		
L	div	Internal 		
L	div	Internal 		
L	mod	Internal 		
L	mod	Internal 		
<b>Context</b>	<b>Implementation</b>			
L	_msgSender	Internal 		
L	_msgData	Internal 		
<b>Address</b>	<b>Library</b>			
L	isContract	Internal 		
L	sendValue	Internal 		
L	functionCall	Internal 		
L	functionCall	Internal 		
L	functionCallWithValue	Internal 		
L	functionCallWithValue	Internal 		





































L	_functionCallWithValue	Private 		
<b>Ownable</b>	<b>Implementation</b>	<b>Context</b>		
L		Public 		NO 
L	owner	Public 		NO 
L	renounceOwnership	Public 		onlyOwner
L	transferOwnership	Public 		onlyOwner
L	geUnlockTime	Public 		NO 
L	lock	Public 		onlyOwner
L	unlock	Public 		NO 
<b>IDEXFactory</b>	<b>Interface</b>			
L	createPair	External 		NO 
<b>IDEXRouter</b>	<b>Interface</b>			
L	factory	External 		NO 
L	WETH	External 		NO 
L	addLiquidity	External 		NO 
L	addLiquidityETH	External 		NO 
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External 		NO 



L	swapExactETHForTokensSupporting FeeOnTransferTokens	External ⚠		NO⚠
L	swapExactTokensForETHSupporting FeeOnTransferTokens	External ⚠		NO⚠
<b>MetaSnails</b>	<b>Implementation</b>	<b>Context, IERC20, Ownable</b>		
L	isBGame	Public ⚠		NO⚠
L	updateBGame	Public ⚠		onlyOwner
L		Public ⚠		NO⚠
L	removeAllFee	Private 		
L	setSaleFee	Private 		
L	restoreAllFee	Private 		
L	setAllBuyFees	Public ⚠		onlyOwner
L	setAllSaleFees	Public ⚠		onlyOwner
L	prepareForPresale	External ⚠		onlyOwner
L	afterPresale	External ⚠		onlyOwner
L	name	Public ⚠		NO⚠
L	symbol	Public ⚠		NO⚠
L	decimals	Public ⚠		NO⚠
L	totalSupply	Public ⚠		NO⚠

L	balanceOf	Public 		NO 
L	transfer	Public 		NO 
L	allowance	Public 		NO 
L	approve	Public 		NO 
L	transferFrom	Public 		NO 
L	increaseAllowance	Public 		NO 
L	decreaseAllowance	Public 		NO 
L	isExcludedFromReward	Public 		NO 
L	totalFees	Public 		NO 
L	deliver	Public 		NO 
L	reflectionFromToken	Public 		NO 
L	tokenFromReflection	Public 		NO 
L	excludeFromReward	Public 		onlyOwner
L	includeInReward	External 		onlyOwner
L	_transferBothExcluded	Private 		
L		External 		NO 
L	_reflectFee	Private 		
L	_getValues	Private 		
L	_getTValues	Private 		



L	_getRValues	Private 		
L	_getRate	Private 		
L	_getCurrentSupply	Private 		
L	_takeLiquidity	Private 		
L	calculateTaxFee	Private 		
L	calculateLiquidityFee	Private 		
L	isExcludedFromFee	Public 		NO 
L	_approve	Private 		
L	_transfer	Private 		
L	swapAndLiquify	Private 		lockTheSwap
L	swapTokensForEth	Private 		
L	addLiquidity	Private 		
L	_tokenTransfer	Private 		
L	manualBurn	Public 		onlyOwner
L	_transferStandard	Private 		
L	_transferToExcluded	Private 		

L	_transferFromExcluded	Private 		
L	setExcludedFromFee	Public 		onlyOwner
L	setExcludedFromWhale	Public 		onlyOwner
L	setMarketingAddress	External 		onlyOwner
L	setMaxSaleLimit	External 		onlyOwner
L	setSwapBackDivisor	External 		onlyOwner
L	setMaxBuyLimit	External 		onlyOwner
L	setSwapAndLiquifyEnabled	Public 		onlyOwner
L	setNumTokensSellToAddToLiquidity	Public 		onlyOwner
L	setMaxWalletLimit	Public 		onlyOwner
L	_beforeTokenTransfer	Internal 		
L	swapBackUpperLimitAmount	Public 		NO 
L	swapBack	Private 		lockTheSwap
L	shouldSwapBack	Private 		lockTheSwap
L	swapETHForTokens	Private 		
L	setBuybackUpperLimit	External 		onlyOwner
L	setBuyBackEnabled	Public 		onlyOwner

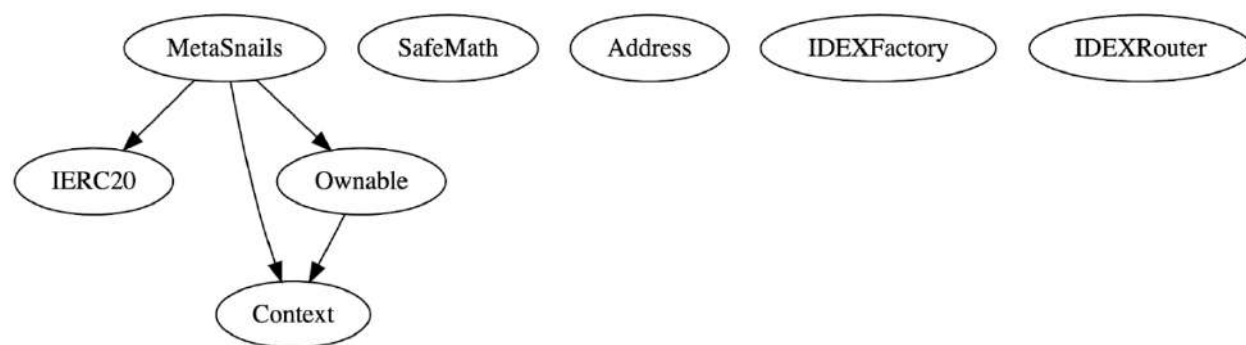


L	manualSwapBack	External !		onlyOwner
L	setFeeVariable	External !		onlyOwner

### Legend

Symbol	Meaning
	Function can modify state
	Function is payable

## Inheritance Hierarchy



# Security issue checking status

- High severity issues

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

```
ftrace | funcSig
function setAllBuyFees(
    uint256 taxFee↑,
    uint256 liquidityFee↑,
    uint256 marketingFee↑
) public onlyOwner {
    _taxFee = taxFee↑;
    _previousTaxFee = taxFee↑;
    _liquidityFee = liquidityFee↑;
    _previousLiquidityFee = liquidityFee↑;
    _marketingFee = marketingFee↑;
    _previousMarketingFee = marketingFee↑;
}

ftrace | funcSig
function setAllSaleFees(
    uint256 taxFee↑,
    uint256 liquidityFee↑,
    uint256 marketingFee↑
) public onlyOwner {
    _saleTaxFee = taxFee↑;
    _saleLiquidityFee = liquidityFee↑;
    _saleMarketingFee = marketingFee↑;
}
```

The owner can change max sell limit without any limit (can set 0)

```
ftrace | funcSig
function setMaxSaleLimit(uint256 amount↑) external onlyOwner {
    _maxSaleLimit = amount↑;
}
```

- **Medium severity issues**

No medium severity issues found

- **Low severity issues**

No low severity issues found

## Owner privileges

- ❖ The owner can change all buy and sell fees

```
ftrace | funcSig
function setAllBuyFees(
    uint256 taxFee↑,
    uint256 liquidityFee↑,
    uint256 marketingFee↑
) public onlyOwner {
    _taxFee = taxFee↑;
    _previousTaxFee = taxFee↑;
    _liquidityFee = liquidityFee↑;
    _previousLiquidityFee = liquidityFee↑;
    _marketingFee = marketingFee↑;
    _previousMarketingFee = marketingFee↑;
}
```

```
ftrace | funcSig
function setAllSaleFees(
    uint256 taxFee↑,
    uint256 liquidityFee↑,
    uint256 marketingFee↑
) public onlyOwner {
    _saleTaxFee = taxFee↑;
    _saleLiquidityFee = liquidityFee↑;
    _saleMarketingFee = marketingFee↑;
}
```

- ❖ The owner can include/exclude wallets from rewards

```
ftrace | funcSig
function excludeFromReward(address account↑) public onlyOwner {
    require(!_isExcluded[account↑], "Account is already excluded");
    if (_rOwned[account↑] > 0) {
        _tOwned[account↑] = tokenFromReflection(_rOwned[account↑]);
    }
    _isExcluded[account↑] = true;
    _excluded.push(account↑);
}

ftrace | funcSig
function includeInReward(address account↑) external onlyOwner {
    require(!_isExcluded[account↑], "Account is already 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;
        }
    }
}
```

- ❖ The owner can manually burn tokens

```
ftrace | funcSig
function manualBurn(uint256 burnAmount↑) public onlyOwner {
    removeAllFee();
    _transferStandard(_owner(), burnAddress, burnAmount↑);
    restoreAllFee();
}
```

- ❖ The owner can include/exclude wallets from fee and whale

```
ftrace | funcSig
function setExcludedFromFee(address account↑, bool _enabled↑)
    public
    onlyOwner
{
    _isExcludedFromFee[account↑] = _enabled↑;
}

ftrace | funcSig
function setExcludedFromWhale(address account↑, bool _enabled↑)
    public
    onlyOwner
{
    _isExcludedFromWhale[account↑] = _enabled↑;
}
```

- ❖ The owner can change marketing address

```
ftrace | funcSig
function setMarketingAddress(address newWallet↑) external onlyOwner {
    _marketingWallet = payable(newWallet↑);
}
```

- ❖ The owner can change max sell limit

```
ftrace | funcSig
function setMaxSaleLimit(uint256 amount↑) external onlyOwner {
    _maxSaleLimit = amount↑;
}
```

- ❖ The owner can change max buy limit

```
ftrace | funcSig
function setMaxBuyLimit(uint256 amount↑) external onlyOwner {
    _maxBuyLimit = amount↑;
}
```



- ❖ The owner can enable/disable swapping

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

- ❖ The owner can change max wallet limit

```
ftrace | funcSig
function setMaxWalletLimit(uint256 amount↑) public onlyOwner {
    maxLimit = amount↑;
}
```

- ❖ The owner can change swap back upper limit and enable/disable buy back

```
ftrace | funcSig
function setBuybackUpperLimit(uint256 swapBackLimit↑) external onlyOwner {
    swapBackUpperLimit = swapBackLimit↑ * 10**18;
}

ftrace | funcSig
function setBuyBackEnabled(bool _enabled↑) public onlyOwner {
    swapBackEnabled = _enabled↑;
    emit SwapBackEnabledUpdated(_enabled↑);
}
```

- ❖ The owner can manually swap back tokens

```
ftrace | funcSig
function manualSwapBack(uint256 amount↑) external onlyOwner {
    swapBack(amount↑ * 10**15);
}
```

- ❖ The owner can enable/disable fees

```
ftrace | funcSig
function setFeeVariable(bool _enableFee↑) external onlyOwner {
    enableFee = _enableFee↑;
}
```



# Audit conclusion

While conducting the audit of the Meta Snails smart contract, it was observed that there is nothing alarming with the code in functional wise and it contains two high severity issues since the owner has substantial control within the ecosystem.