



# **RugFreeCoins Audit**



**Meta Royal Token**

**Smart Contract Security Audit**

**February 28, 2022**



# Contents

Audit details	1
Disclaimer	2
Background	3
About the project	4
Target market and the concept	5
Potential to grow with score points	6
Total Points	6
Contract details	7
Contract code function details	8
Contract description table	10
Security issue checking status	16
Audit conclusion	19

# Audit details



**Audited project**  
Meta Royal Token



**Contract Address**  
0xc6F6D16C72f705eC983ED932604068872DC7f889



**Client contact**  
Meta Royal Team



**Blockchain**  
Binance smart chain



**Project website**  
<https://www.metaroyale.app/>

# Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Rugfreecoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Rugfreecoins) owe no duty of care towards you or any other person, nor does Rugfreecoins make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Rugfreecoins hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Rugfreecoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Rugfreecoins, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

# Background

Rugfreecoins was commissioned by the Meta Royale Team to perform an audit of the smart contract.

**<https://bscscan.com/token/0xc6F6D16C72f705eC983ED932604068872DC7f889>**

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 Royale is a token built on the Binance Smart Chain that is with an innovative investment use case the main purpose of which is to seek out constant revenue sources, which in turn, powers reward combined with the play to earn game. Each transaction, purchase, and sale incur a 10% fee.

## Features

- The **sustainability fee of 7% when buying and selling for marketing** is what allows Meta Royale to hold the aforementioned promise. Tokens will be swapped into BUSD and will be sent to a marketing wallet per transaction. This way, Meta Royale 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 3% from buying and selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

## Tokenomics

### 10% fee when buying and selling

- 7% of trade goes to the marketing wallet in BUSD.
- 3% of trade goes to the liquidity pool.

# Target market and the concept

## Target market

- Anyone who's interested in the Crypto space with long-term investment plans.
- Anyone who's ready to earn a passive income in BNB by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in collecting NFTs or trading NFTs.
- Anyone who's interested in playing the meta royale game and winning tokens.
- Anyone who's interested in taking part with the future plans of the Meta Royale token.
- Anyone who's interested in making financial transactions with any other party using Meta Royale as the currency.

## Core concept

### Sustainable mechanism

The **sustainability fee of 7% when buying and selling for marketing** is what allows Meta Royale to promote the token and use funds to further the development of the platform. Tokens will be swapped into BUSD and will be sent to a marketing wallet per transaction. This way, Meta Royale 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 3%**, 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	7/10
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		<b>8.5/10</b>



# Contract details

## Token contract details for 28<sup>th</sup> February 2022

Contract name	MetaRoyale
Contract address	0xc6F6D16C72f705eC983ED932604068872DC7f889
Token supply	300,000,000
Token ticker	MRVR
Decimals	9
Token holders	1
Transaction count	2
Marketing wallet	0x572d8a7e25688ba4db2e68e99b83f7ad6c07da07
Contract deployer address	0x97361F6979756A647458DC3EAAB802Db416997a3
Contract's current owner address	0x2f505cc8bb6985638446ad48b3534f5b2085fa76











# 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

<b>7</b>	<b>Transaction ordering dependence</b>		<b>pass</b>
<b>8</b>	<b>Block properties dependence</b>		<b>pass</b>
<b>9</b>	<b>Pseudo random number generator (PRNG)</b>		<b>pass</b>
<b>10</b>	<b>DoS (Denial of Service)</b>		<b>pass</b>
<b>11</b>	<b>Token vesting implementation</b>		<b>pass</b>
<b>12</b>	<b>Fake deposit</b>		<b>pass</b>
<b>13</b>	<b>Event security</b>		<b>pass</b>















# Contract description table












The below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions, and implementations with their visibility and mutability.
















Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
<b>MetaRoyale</b>	<b>Implementation</b>	<b>IERC20, Ownable</b>		
L		Public !		NO !
L		External !		NO !
L	totalSupply	External !		NO !
L	name	Public !		NO !
L	symbol	Public !		NO !
L	decimals	Public !		NO !
L	balanceOf	Public !		NO !
L	allowance	External !		NO !
L	approve	Public !		NO !
L	_approve	Internal 		
L	approveMax	External !		NO !
L	transfer	External !		NO !
L	transferFrom	External !		NO !
L	_transferFrom	Internal 		



L	_basicTransfer	Internal 		
L	shouldTakeFee	Internal 		
L	takeFee	Internal 		
L	shouldSwapBack	Internal 		
L	clearStuckBalance	External !		onlyOwner
L	updateUniswapRouter	Public !		onlyOwner
L	updateBuyFees	Public !		onlyOwner
L	updateSellFees	Public !		onlyOwner
L	changeSwapAmount	Public !		onlyOwner
L	setGetFeesOnNormalTx	Public !		onlyOwner
L	swapBackInBnb	Internal 		swapping
L	swapAndLiquify	Private 		
L	swapTokensForEth	Private 		
L	swapTokensForTokens	Private 		
L	addLiquidity	Private 		
L	setIsFeeExempt	External !		onlyOwner
L	setFeeReceivers	External !		onlyOwner
L	setSwapBackSettings	External !		onlyOwner
<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	_transferOwnership	Internal 		
<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	tryAdd	Internal 		
L	trySub	Internal 		
L	tryMul	Internal 		
L	tryDiv	Internal 		
L	tryMod	Internal 		
L	add	Internal 		
L	sub	Internal 		

L	mul	Internal 		
L	div	Internal 		
L	mod	Internal 		
L	sub	Internal 		
L	div	Internal 		
L	mod	Internal 		
<b>IUniswapV2Router02</b>	<b>Interface</b>	<b>IUniswapV2Router01</b>		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External !		NO !
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External !		NO !
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !		NO !
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External !		NO !
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !		NO !
<b>IUniswapV2Factory</b>	<b>Interface</b>			
L	feeTo	External !		NO !
L	feeToSetter	External !		NO !
L	getPair	External !		NO !
L	allPairs	External !		NO !
L	allPairsLength	External !		NO !

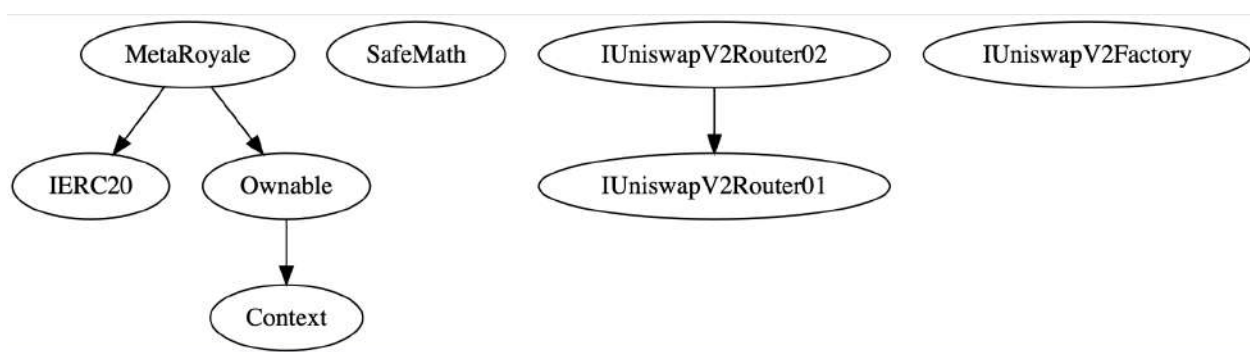
L	createPair	External !		NO !
L	setFeeTo	External !		NO !
L	setFeeToSetter	External !		NO !
<b>Context</b>	<b>Implementation</b>			
L	_msgSender	Internal 		
L	_msgData	Internal 		
<b>IUniswapV2Router01</b>	<b>Interface</b>			
L	factory	External !		NO !
L	WETH	External !		NO !
L	addLiquidity	External !		NO !
L	addLiquidityETH	External !		NO !
L	removeLiquidity	External !		NO !
L	removeLiquidityETH	External !		NO !
L	removeLiquidityWithPermit	External !		NO !
L	removeLiquidityETHWithPermit	External !		NO !
L	swapExactTokensForTokens	External !		NO !
L	swapTokensForExactTokens	External !		NO !
L	swapExactETHForTokens	External !		NO !
L	swapTokensForExactETH	External !		NO !

L	swapExactTokensForETH	External !		NO !
L	swapETHForExactTokens	External !		NO !
L	quote	External !		NO !
L	getAmountOut	External !		NO !
L	getAmountIn	External !		NO !
L	getAmountsOut	External !		NO !
L	getAmountsIn	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 low severity issues found

## Owner privileges

- ❖ The owner can get contract BNB balance to owner wallet

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

- ❖ The owner can update the router address.

```
ftrace | funcSig
function updateUniswapRouter(address newAddress↑) public onlyOwner {
    require(
        newAddress↑ != address(router),
        "The router already has that address"
    );

    router = IUniswapV2Router02(newAddress↑);
    pair = IUniswapV2Factory(router.factory()).createPair(
        WBNB,
        address(this)
    );
}
```



- ❖ The owner can update all buy and sell fees maximum of up to 25%.

```
ftrace | funcSig
function updateBuyFees(uint256 marketing↑, uint256 liquidity↑)
    public
    onlyOwner
{
    buyLiquidityFee = liquidity↑;
    buyMarketingFee = marketing↑;
    buyTotalFees = marketing↑.add(liquidity↑);
    require(buyTotalFees <= 25, "Total Fees can not be grater than 25%");
}

ftrace | funcSig
function updateSellFees(uint256 marketing↑, uint256 liquidity↑)
    public
    onlyOwner
{
    sellLiquidityFee = liquidity↑;
    sellMarketingFee = marketing↑;
    sellTotalFees = marketing↑.add(liquidity↑);
    require(sellTotalFees <= 25, "Total Fees can not be grater than 25%");
}
```

- ❖ The owner can change the swap percentage.

```
// change swap percentage
ftrace | funcSig
function changeSwapAmount(uint256 _marketingSwap↑, uint256 _liquiditySwap↑)
    public
    onlyOwner
{
    marketingSwap = _marketingSwap↑;
    liquiditySwap = _liquiditySwap↑;
    uint256 totalSwap = _marketingSwap↑.add(_liquiditySwap↑);

    require(totalSwap == 100, "Total swap percentage should equal to 100%");
}
```

- ❖ The owner can enable/disable fees on normal transactions.

```
ftrace | funcSig
function setGetFeesOnNormalTx(bool _status↑) public onlyOwner {
    isGetNormalFees = _status↑;
}
```

- ❖ The owner can include/exclude wallets from the fee.

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

- ❖ The owner can change the marketing wallet address.

```
ftrace | funcSig
function setFeeReceivers(address _wallet↑) external onlyOwner {
    marketingWallet = _wallet↑;
}
```

- ❖ The owner can enable/disable swap back and can change swap point.

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

# Audit conclusion

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