RugFreeCoins Audit

The Willy Token
Smart Contract Security Audit

May 25th ,2023

Contents

Audit details

Audited project
Willy Token

Contract Address
0xd074Ec0953F0456a48Cbddb55157ada884DD07e6

Client contact
Willy Team

Blockchain
Binance smart chain

Project website
https://willybsc.xyz/

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.

Overview

✅ No mint function found, the owner cannot mint tokens after initial deployment.

✅ The owner can’t set a max transaction limit

✅ The owner can’t pause trading.

✅ The owner can't set fees over 25%.

✅ The owner can't blacklist wallets.

✅ The owner can’t set a max wallet limit

✅ The owner can’t claim the contract’s balance of its own token.

Background

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

https://bscscan.com/token/0xd074Ec0953F0456a48Cbddb55157ada884DD07e6

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, and long-term sustainability, and as a guide to improving the smart contract's security posture by remediating the identified issues.

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 by holding tokens.
- Anyone who’s interested in trading tokens.
- Anyone who’s interested in taking part of the Willy ecosystem.
- Anyone who’s interested in taking part in the future plans of Willy Token.
- Anyone who’s interested in making financial transactions with any other party using Willy Token as the currency.

Potential to grow with score points

1.	Project efficiency	7/10
2.	Project uniqueness	6/10
3	Information quality	6/10
4	Service quality	7/10
5	System quality	7/10
6	Impact on the community	7/10
7	Impact on the business	7/10
8	Preparing for the future	7/10
9	Smart contract security	10/10
10	Smart contract functionality assessment	10/10
Total Points		**7.4/10**

Contract details

Token contract details for 25th of May 2023

Contract name	Willy
Contract address	0xd074Ec0953F0456a48Cbddb55157ada884DD07e6
Token supply	10,000,000,000
Token ticker	WILLY
Decimals	18
Token holders	1
Transaction count	1
Contract deployer address	0x687780c1493173179c2368D7B557e0A77c5C3a06
Contract's current owner address	0x687780c1493173179c2368D7B557e0A77c5C3a06

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	
		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

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
WILLY	**Implementation**	**IERC20Metadata , AccessControl**		
L		Public !	🛑	NO !
L	balanceOf	Public !		NO !
L	transfer	External !	🛑	NO !
L	allowance	Public !		NO !
L	approve	External !	🛑	NO !
L	transferFrom	External !	🛑	NO !
L	increaseAllowance	External !	🛑	NO !
L	decreaseAllowance	External !	🛑	NO !
L	setLpToken	External !	🛑	onlyRole
L	setExcludedFromFee	Public !	🛑	onlyRole
L	setExcludedFromSwap	Public !	🛑	onlyRole
L	setRewardSwapReceivers	External !	🛑	onlyRole
L	setRewardSellRate	External !	🛑	onlyRole
L	setRewardBuyRate	External !	🛑	onlyRole

L	resetRewardsAmount	External !	🛑	onlyRole
L	updateBuyRates	External !	🛑	onlyRole
L	updateSellRates	External !	🛑	onlyRole
L	updateTransferRates	External !	🛑	onlyRole
L	resetCounter	External !	🛑	onlyRole
L	setLimit	External !	🛑	onlyRole
L	updateBurnFeeReceivers	External !	🛑	onlyRole
L	updateLiquidityFeeReceivers	External !	🛑	onlyRole
L	resetLiquidityFee	External !	🛑	onlyRole
L	updateSwapFeeReceivers	External !	🛑	onlyRole
L	resetSwapFee	External !	🛑	onlyRole
L	setEnabledSwapForSell	External !	🛑	onlyRole
L	_transfer	Internal 🔒	🛑	
L	_takeFees	Internal 🔒	🛑	
L	_transferAmount	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_approve	Internal 🔒	🛑	
L	_setRouterAndPair	Internal 🔒	🛑	
L	_calcFee	Internal 🔒		
L	_isSell	Internal 🔒		
L	_isBuy	Internal 🔒		

L	_swapTokensForToken1	Internal 🔒	🛑	lockThe Swap
L	_addLiquidity	Internal 🔒	🛑	lockThe Swap
IERC20 Metadata	**Interface**	**IERC20**		
L	name	External ❗		NO ❗
L	symbol	External ❗		NO ❗
L	decimals	External ❗		NO ❗
IERC20	**Interface**			
L	totalSupply	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	transfer	External ❗	🛑	NO ❗
L	allowance	External ❗		NO ❗
L	approve	External ❗	🛑	NO ❗
L	transferFrom	External ❗	🛑	NO ❗
Access Control	**Implementation**	**Context, IAccessControl, ERC165**		
L	supportsInterface	Public ❗		NO ❗
L	hasRole	Public ❗		NO ❗
L	_checkRole	Internal 🔒		
L	_checkRole	Internal 🔒		

L	getRoleAdmin	Public !		NO !
L	grantRole	Public !	🛑	onlyRole
L	revokeRole	Public !	🛑	onlyRole
L	renounceRole	Public !	🛑	NO !
L	_setupRole	Internal 🔒	🛑	
L	_setRoleAdmin	Internal 🔒	🛑	
L	_grantRole	Internal 🔒	🛑	
L	_revokeRole	Internal 🔒	🛑	
IAccessControl	**Interface**			
L	hasRole	External !		NO !
L	getRoleAdmin	External !		NO !
L	grantRole	External !	🛑	NO !
L	revokeRole	External !	🛑	NO !
L	renounceRole	External !	🛑	NO !
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
Strings	**Library**			
L	toString	Internal 🔒		

L	toHexString	Internal 🔒		
L	toHexString	Internal 🔒		
L	toHexString	Internal 🔒		
Math	**Library**			
L	max	Internal 🔒		
L	min	Internal 🔒		
L	average	Internal 🔒		
L	ceilDiv	Internal 🔒		
L	mulDiv	Internal 🔒		
L	mulDiv	Internal 🔒		
L	sqrt	Internal 🔒		
L	sqrt	Internal 🔒		
L	log2	Internal 🔒		
L	log2	Internal 🔒		
L	log10	Internal 🔒		
L	log10	Internal 🔒		
L	log256	Internal 🔒		
L	log256	Internal 🔒		
ERC165	**Implementation**	**IERC165**		
L	supportsInterface	Public !		NO !

IERC165	**Interface**			
L	supportsInterface	External !		NO !
IUniswapV2 Router02	**Interface**	**IUniswapV2 Router01**		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External !		NO !
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External !		NO !
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !		NO !
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External !		NO !
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !		NO !
IUniswapV2 Router01	**Interface**			
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 !

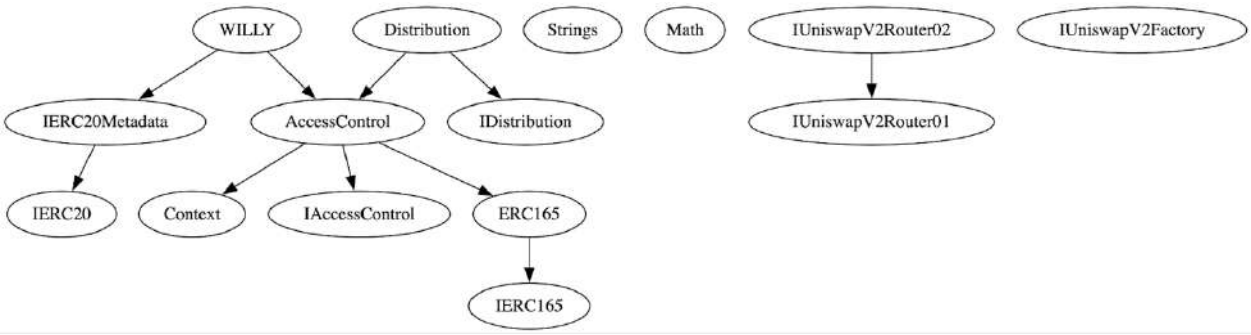
L	swapExactETHForTokens	External !		NO !
L	swapTokensForExactETH	External !		NO !
L	swapExactTokensForETH	External !		NO !
L	swapETHForExactTokens	External !		NO !
L	quote	External !		NO !
L	getAmountOut	External !		NO !
L	getAmountIn	External !		NO !
L	getAmountsOut	External !		NO !
L	getAmountsIn	External !		NO !
IUniswapV2 Factory	**Interface**			
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 !
Distribution	**Implementation**	**IDistribution, AccessControl**		

L		Public !	🛑	NO !
L	recoverTokensFor	External !	🛑	onlyRole
IDistribution	**Interface**			
L	recoverTokensFor	External !	🛑	NO !

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy



Security issue checking status

- ❖ **High severity issues**

Out of Gas Issue

When sending collected fees and LP tokens, the owner can add an unlimited number of fee receiver wallets without any limitation. However, if the owner adds too many fee receiver wallets, the contract transaction may fail due to gas limitations. Consequently, users will not be able to buy, sell, or transfer, and they might be required to pay a substantial amount of gas fees for simple transactions.

Informed & Fixed

```
ftrace | funcSig
function setRewardSwapReceivers(
    address[] calldata _rewardSwapReceivers,
    uint256[] calldata _rewardSwapReceiversRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _rewardSwapReceivers.length == _rewardSwapReceiversRate.length,
        "size"
    );

    uint256 totalRate = 0;
    for (uint256 i = 0; i < _rewardSwapReceiversRate.length; i++) {
        totalRate += _rewardSwapReceiversRate[i];
    }
    require(totalRate == 10000, "rate");

    delete rewardSwapReceivers;
    delete rewardSwapReceiversRate;

    for (uint i = 0; i < _rewardSwapReceivers.length; i++) {
        rewardSwapReceivers.push(_rewardSwapReceivers[i]);
        rewardSwapReceiversRate.push(_rewardSwapReceiversRate[i]);
    }

    emit RewardSwapReceiversUpdated(
        _rewardSwapReceivers,
        _rewardSwapReceiversRate
    );
}
```

```
ftrace | funcSig
function updateLiquidityFeeReceivers(
    address[] calldata _liquidityFeeReceivers,
    uint256[] calldata _liquidityFeeReceiversRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _liquidityFeeReceivers.length == _liquidityFeeReceiversRate.length,
        "size"
    );

    uint256 totalRate = 0;
    for (uint256 i = 0; i < _liquidityFeeReceiversRate.length; i++) {
        totalRate += _liquidityFeeReceiversRate[i];
    }
    require(totalRate == 10000, "rate");

    delete liquidityFeeReceivers;
    delete liquidityFeeReceiversRate;

    for (uint i = 0; i < _liquidityFeeReceivers.length; i++) {
        liquidityFeeReceivers.push(_liquidityFeeReceivers[i]);
        liquidityFeeReceiversRate.push(_liquidityFeeReceiversRate[i]);
    }

    emit LiquidityFeeReceiversUpdated(
        _liquidityFeeReceivers,
        _liquidityFeeReceiversRate
    );
}
```

When buying and selling, the system calculates a reward fee; however, this fee is not collected. When swapping, the system attempts to deduct the collected tokens from the swap wallet.

Informed & Fixed

```
uint256 rewardBuyToSwap = rewardBuyAmount + rewardSellAmount;
if (
    rewardBuyToSwap > 0 &&
    balanceOf(rewardSwapAddress) >= rewardBuyToSwap
) {
    _transferAmount(
        rewardSwapAddress,
        address(this),
        rewardBuyToSwap
    );
    amountToSwap += rewardBuyToSwap;
}
```

```
        rewardBuyAmount += _calcFee(resultAmount, rewardBuyRate);
    } else if (_isSell(_from, _to)) {
        burnFeeRes = _calcFee(resultAmount, burnFeeSellRate);
        liquidityFeeRes = _calcFee(
            resultAmount,
            liquidityFeeSellRate
        );
        swapFeeRes = _calcFee(resultAmount, swapFeeSellRate);

        rewardSellAmount += _calcFee(resultAmount, rewardSellRate);
    } else {
```

The owner has the ability to add multiple wallets to receive the burn fee. However, when tokens are burned, they should be sent to the null or dead address. If tokens are sent to any address other than the null or dead address, they will not be burned, indicating that it is not an actual burn.

Informed & Fixed

```
ftrace | funcSig
function updateBurnFeeReceivers(
    address[] calldata _burnFeeReceivers,
    uint256[] calldata _burnFeeReceiversRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _burnFeeReceivers.length == _burnFeeReceiversRate.length,
        "size"
    );

    uint256 totalRate = 0;
    for (uint256 i = 0; i < _burnFeeReceiversRate.length; i++) {
        totalRate += _burnFeeReceiversRate[i];
    }
    require(totalRate == 10000, "rate");

    delete burnFeeReceivers;
    delete burnFeeReceiversRate;

    for (uint i = 0; i < _burnFeeReceivers.length; i++) {
        burnFeeReceivers.push(_burnFeeReceivers[i]);
        burnFeeReceiversRate.push(_burnFeeReceiversRate[i]);
    }

    emit BurnFeeReceiversUpdated(_burnFeeReceivers, _burnFeeReceiversRate);
}
```

- **Medium severity issues**
  No medium severity issues found

- **Low severity issues**
  No low-severity issues found

- **Centralization Risk**
  No centralization issues found

Owner privileges

- Owner can add/remove LP token address

```
ftrace | funcSig
function setLpToken(
    address _lpToken,
    bool _lp
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(_lpToken != address(0), "BEP20: invalid LP address");
    require(_lpToken != pair, "ERC20: exclude default pair");

    isLpToken[_lpToken] = _lp;

    emit LpTokenUpdated(_lpToken, _lp);
}
```

- Owner can enable/disable wallets from fees

```
ftrace | funcSig
function setExcludedFromFee(
    address _address,
    bool _isExcludedFromFee
) public onlyRole(DEFAULT_ADMIN_ROLE) {
    excludedFromFee[_address] = _isExcludedFromFee;

    emit ExcludedFromFee(_address, _isExcludedFromFee);
}
```

❖ Owner can add reward swap receiver address and rates

```
ftrace | funcSig
function setRewardSwapReceivers(
    address[] calldata _rewardSwapReceivers,
    uint256[] calldata _rewardSwapReceiversRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _rewardSwapReceivers.length == _rewardSwapReceiversRate.length,
        "size"
    );

    uint256 totalRate = 0;
    for (uint256 i = 0; i < _rewardSwapReceiversRate.length; i++) {
        totalRate += _rewardSwapReceiversRate[i];
    }
    require(totalRate == 10000, "rate");

    delete rewardSwapReceivers;
    delete rewardSwapReceiversRate;

    for (uint i = 0; i < _rewardSwapReceivers.length; i++) {
        rewardSwapReceivers.push(_rewardSwapReceivers[i]);
        rewardSwapReceiversRate.push(_rewardSwapReceiversRate[i]);
    }

    emit RewardSwapReceiversUpdated(
        _rewardSwapReceivers,
        _rewardSwapReceiversRate
    );
}
```

❖ Owner can change sell reward rate maximum up to 30%

```
ftrace | funcSig
function setRewardSellRate(
    uint256 _rewardSellRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(_rewardSellRate <= 3000, "_rewardSellRate");
    // min: 0%; max: 30%
    rewardSellRate = _rewardSellRate;

    emit RewardSellRateUpdated(_rewardSellRate);
}
```

- Owner can change buy reward rate maxim up to 30%

```
ftrace | funcSig
function setRewardBuyRate(
    uint256 _rewardBuyRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(_rewardBuyRate <= 3000, "_rewardBuyRate");
    // min: 0%; max: 30%
    rewardBuyRate = _rewardBuyRate;

    emit RewardBuyRateUpdated(_rewardBuyRate);
}
```

- Owner can reset buy and sell collected reward token amounts

```
ftrace | funcSig
function resetRewardsAmount() external onlyRole(DEFAULT_ADMIN_ROLE) {
    rewardSellAmount = 0;
    rewardBuyAmount = 0;

    emit RewardsAmountReseted();
}
```

- Owner can change buy fees maximum upto 9%

```
ftrace | funcSig
function updateBuyRates(
    uint256 _burnFeeBuyRate,
    uint256 _liquidityFeeBuyRate,
    uint256 _swapFeeBuyRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _burnFeeBuyRate + _liquidityFeeBuyRate + _swapFeeBuyRate <= 900,
        "rate"
    );

    burnFeeBuyRate = _burnFeeBuyRate;
    liquidityFeeBuyRate = _liquidityFeeBuyRate;
    swapFeeBuyRate = _swapFeeBuyRate;

    emit BuyFeesUpdated(
        _burnFeeBuyRate,
        _liquidityFeeBuyRate,
        _swapFeeBuyRate
    );
}
```

❖ Owner can change sell fees maximum up to 9%

```
ftrace | funcSig
function updateSellRates(
    uint256 _burnFeeSellRate,
    uint256 _liquidityFeeSellRate,
    uint256 _swapFeeSellRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _burnFeeSellRate + _liquidityFeeSellRate + _swapFeeSellRate <= 900,
        "rate"
    );

    burnFeeSellRate = _burnFeeSellRate;
    liquidityFeeSellRate = _liquidityFeeSellRate;
    swapFeeSellRate = _swapFeeSellRate;

    emit SellFeesUpdated(
        _burnFeeSellRate,
        _liquidityFeeSellRate,
        _swapFeeSellRate
    );
}
```

❖ Owner can change transfer fee rate maximum up to 9%

```
ftrace | funcSig
function updateTransferRates(
    uint256 _burnFeeTransferRate,
    uint256 _liquidityFeeTransferRate,
    uint256 _swapFeeTransferRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _burnFeeTransferRate +
            _liquidityFeeTransferRate +
            _swapFeeTransferRate <=
            900,
        "rate"
    );

    burnFeeTransferRate = _burnFeeTransferRate;
    liquidityFeeTransferRate = _liquidityFeeTransferRate;
    swapFeeTransferRate = _swapFeeTransferRate;

    emit TransferFeesUpdated(
        _burnFeeTransferRate,
        _liquidityFeeTransferRate,
        _swapFeeTransferRate
    );
}
```

❖ Owner can reset fee counter

```
ftrace | funcSig
function resetCounter() external onlyRole(DEFAULT_ADMIN_ROLE) {
    feeCounter = 0;

    emit FeeCounterReseted();
}
```

❖ Owner can change Burn fee receivers with rates

```
ftrace | funcSig
function updateBurnFeeReceivers(
    address[] calldata _burnFeeReceivers↑,
    uint256[] calldata _burnFeeReceiversRate↑
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _burnFeeReceivers↑.length == _burnFeeReceiversRate↑.length,
        "size"
    );

    uint256 totalRate = 0;
    for (uint256 i = 0; i < _burnFeeReceiversRate↑.length; i++) {
        totalRate += _burnFeeReceiversRate↑[i];
    }
    require(totalRate == 10000, "rate");

    delete burnFeeReceivers;
    delete burnFeeReceiversRate;

    for (uint i = 0; i < _burnFeeReceivers↑.length; i++) {
        burnFeeReceivers.push(_burnFeeReceivers↑[i]);
        burnFeeReceiversRate.push(_burnFeeReceiversRate↑[i]);
    }

    emit BurnFeeReceiversUpdated(_burnFeeReceivers↑, _burnFeeReceiversRate↑);
}
```

❖ Owner can add liquidity fee receivers with rate

```
ftrace | funcSig
function updateLiquidityFeeReceivers(
    address[] calldata _liquidityFeeReceivers↑,
    uint256[] calldata _liquidityFeeReceiversRate↑
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _liquidityFeeReceivers↑.length == _liquidityFeeReceiversRate↑.length,
        "size"
    );

    uint256 totalRate = 0;
    for (uint256 i = 0; i < _liquidityFeeReceiversRate↑.length; i++) {
        totalRate += _liquidityFeeReceiversRate↑[i];
    }
    require(totalRate == 10000, "rate");

    delete liquidityFeeReceivers;
    delete liquidityFeeReceiversRate;

    for (uint i = 0; i < _liquidityFeeReceivers↑.length; i++) {
        liquidityFeeReceivers.push(_liquidityFeeReceivers↑[i]);
        liquidityFeeReceiversRate.push(_liquidityFeeReceiversRate↑[i]);
    }

    emit LiquidityFeeReceiversUpdated(
        _liquidityFeeReceivers↑,
        _liquidityFeeReceiversRate↑
    );
}
```

❖ Owner can add swap fee receivers with rates

```
ftrace | funcSig
function updateSwapFeeReceivers(
    address[] calldata _swapFeeReceivers,
    uint256[] calldata _swapFeeReceiversRate
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(
        _swapFeeReceivers.length == _swapFeeReceiversRate.length,
        "size"
    );

    uint256 totalRate = 0;
    for (uint256 i = 0; i < _swapFeeReceiversRate.length; i++) {
        totalRate += _swapFeeReceiversRate[i];
    }
    require(totalRate == 10000, "rate");

    delete swapFeeReceivers;
    delete swapFeeReceiversRate;

    for (uint i = 0; i < _swapFeeReceivers.length; i++) {
        swapFeeReceivers.push(_swapFeeReceivers[i]);
        swapFeeReceiversRate.push(_swapFeeReceiversRate[i]);
    }

    emit SwapFeeReceiversUpdated(_swapFeeReceivers, _swapFeeReceiversRate);
}
```

❖ Owner can enable/disable swap when selling

```
ftrace | funcSig
function setEnabledSwapForSell(
    bool _enabledSwapForSell
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    enabledSwapForSell = _enabledSwapForSell;

    emit EnabledSwapForSellUpdated(_enabledSwapForSell);
}
```

Audit conclusion

RugFreeCoins team has performed in-depth testings, line-by-line manual code review, and automated audit of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status: **PASS**

Number of risk issues: **0**

Solidity code functional issue level: **PASS**

Number of owner privileges: **14**

Centralization risk correlated to the active owner: **HIGH**

Smart contract active ownership: **ACTIVE**