RugFreeCoins Audit

JRR Token Audit

Smart Contract Security Audit

September 10, 2021

Contents

Audit details

Audited project

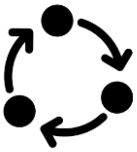
JRR Token

Contract Address

0x8d46739bb6ad55ae438d921cb130afb27e74b46e

Client contact

JRR Token Team



Blockchain

Binance smart chain

Project website

https://www.thetokenofpower.com/

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 JRR Token to perform an audit of the smart contract.

https://bscscan.com/token/0x8d46739bb6ad55ae438d921cb130afb27e74b46e

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

JRR token is a token built on the Binance Smart Chain. Each transaction, purchase and sale incur a fee of 9%.

Features

- Investors can accumulate more JRR tokens by just holding as the smart contract automatically distributes 3% of every transaction tax amongst holders.

- 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.

- 3% is sent to a wallet for marketing and charity. Having the resources necessary to deliver a meticulously marketed project, such as professional branding, paid influencers, charity and donations JRR token the ability for substantial growth.

Tokenomics

9% fee when buying & selling

- 3% of trade goes to holders pockets in token.
- 3% of trade goes to the liquidity pool.
- 3% of trade goes to marketing and charity.

Roadmap


ROAD MAP
Q2 2021
CORE TEAM DEVELOPMENT
SOCIAL MEDIA ACCOUNTS
WEBSITE LAUNCH
MARKETING CAMPAIGN
DX SALE LISTING
PANCAKESWAP LISTING
Q3 2021
NFT LAUNCH
EXCHANGE LISTINGS
INTERVIEWS & INFLUENCERS
CHARITY CONTRIBUTIONS
Q4 2021
INCREASED MARKETING PUSH
SECOND NFT RELEASE

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 JR tokens by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in taking part with the future plans of the project.
- Anyone who's interested in taking part with the good cause of charity of the project.
- Anyone who's interested in collecting NFTs or trading NFTs.
- Anyone who's interested in making financial transactions with any other party using JRR token as the currency.

Core concept

The reward system

3% of each transaction when buying and selling gets sent amongst all holders in tokens. The holders will be eligible to receive tokens, whenever a transaction occurs, 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.

The **fee of 3% development & charity** is what allows JRR to promote the token and use funds to further development of the platform.

Good cause

Charity donations to an educational foundation and multiple environment charities.

JRR NFTs

The future plan

Potential to grow with score points

1.	Project efficiency	8/10
2.	Project uniqueness	8/10
3	Information quality	10/10
4	Service quality	9/10
5	System quality	8/10
6	Impact on the community	10/10
7	Impact on the business	8/10
8	Preparing for the future	8/10
Total Points		**8.63/10**

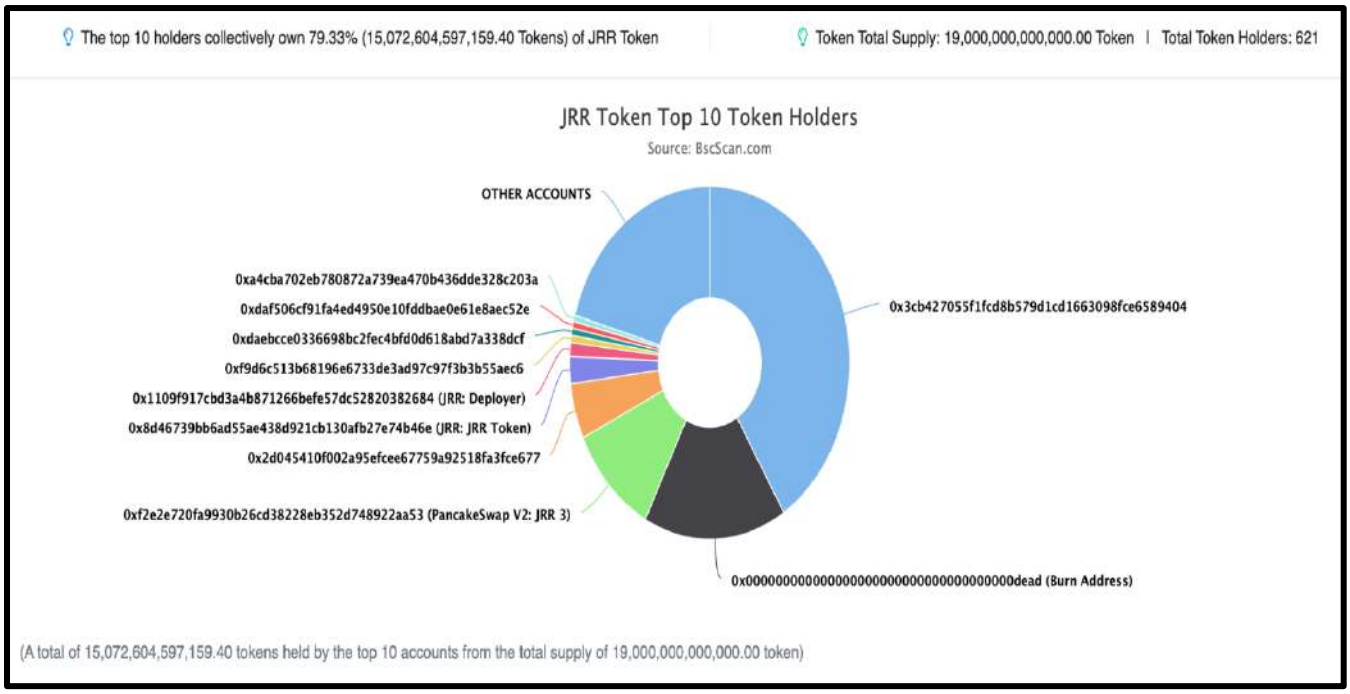
Contract details

Token contract details for 10th September 2021

Contract name	JRR Token
Contract address	0x8d46739bb6ad55ae438d921cb130afb27e74b46e
Token supply	19,000,000,000,000
Token ticker	JRR
Decimals	9
Token holders	621
Transaction count	2,037
Dev wallet address	0x1109f917cbd3a4b871266befe57dc52820382684
Contract deployer address	0x1109f917CbD3a4B871266bEfE57dC52820382684
Contract's current owner address	0x1109f917cbd3a4b871266befe57dc52820382684

Top token holders

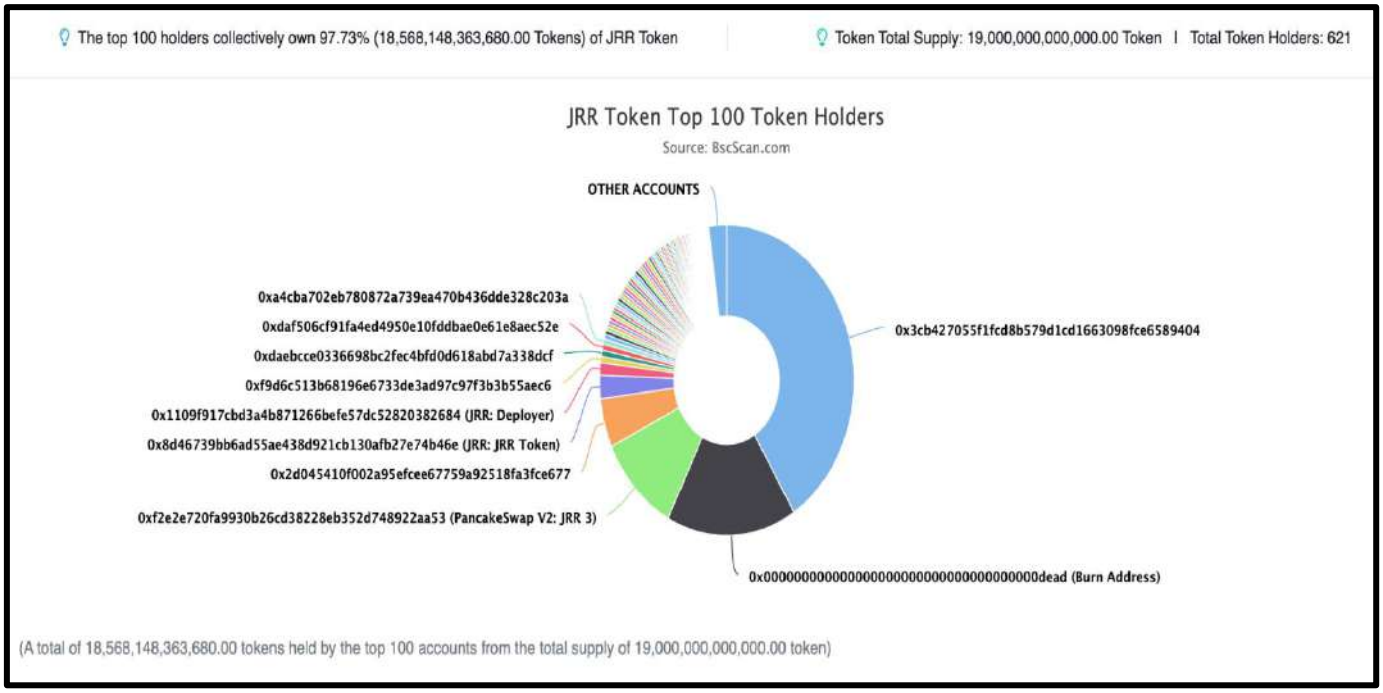
Top 10 Token Holders



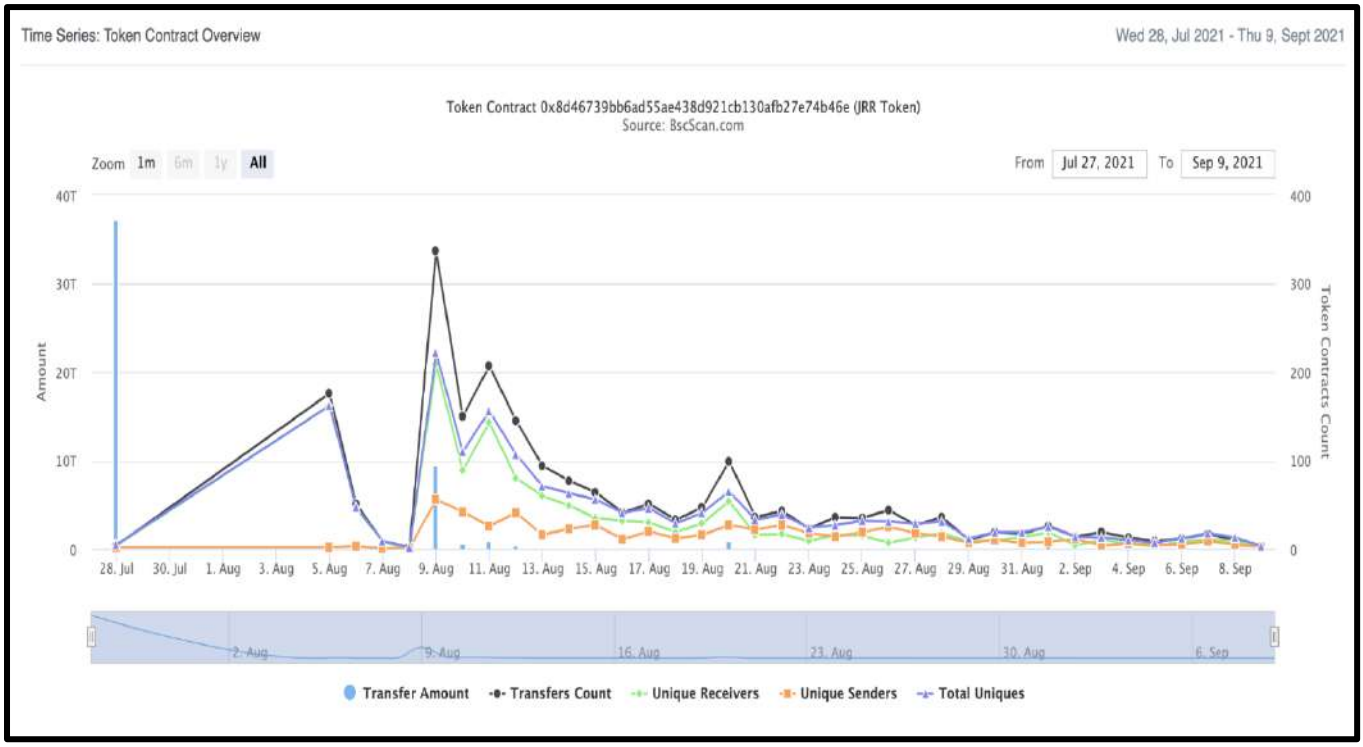
Rank	Address	Quantity (Token)	Percentage
1	0x3cb427055f1fcd8b579d1cd1663098fce6589404	7,818,492,512,470.702270142	41.1500%
2	Burn Address	3,130,017,270,267.79238245	16.4738%
3	PancakeSwap V2: JRR 3	1,972,079,568,107.088014468	10.3794%
4	0x2d045410f002a95efcee67759a92518fa3fce677	967,828,339,644.512500333	5.0938%
5	JRR: JRR Token	466,059,652,999.652410194	2.4529%
6	JRR: Deployer	243,403,242,430.05277084	1.2811%
7	0xf9d6c513b68196e6733de3ad97c97f3b3b55aec6	125,157,190,626.237543792	0.6587%
8	0xdaebcce0336698bc2fec4bfd0d618abd7a338dcf	121,185,102,027.572995743	0.6378%
9	0xdaf506cf91fa4ed4950e10fddbae0e61e8aec52e	120,368,125,995.09836935	0.6335%
10	0xa4cba702eb780872a739ea470b436dde328c203a	108,013,592,590.603715584	0.5685%

Token distribution

Top 100 Token Holders



Contract interaction details



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	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		
└	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❗
SafeMath	**Library**			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		

└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Address	**Library**			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	

└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	_verifyCallResult	Private 🔐		
Ownable	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	lock	Public ❗	🛑	onlyOwner
└	unlock	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❗
└	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❗
CoinToken	**Implementation**	**Context, IERC20, Ownable**		
└		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❗
└	isExcludedFromReward	Public ❗		NO❗
└	totalFees	Public ❗		NO❗
└	deliver	Public ❗	🛑	NO❗
└	reflectionFromToken	Public ❗		NO❗
└	tokenFromReflection	Public ❗		NO❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	_transferBothExcluded	Private 🔐	🛑	
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner

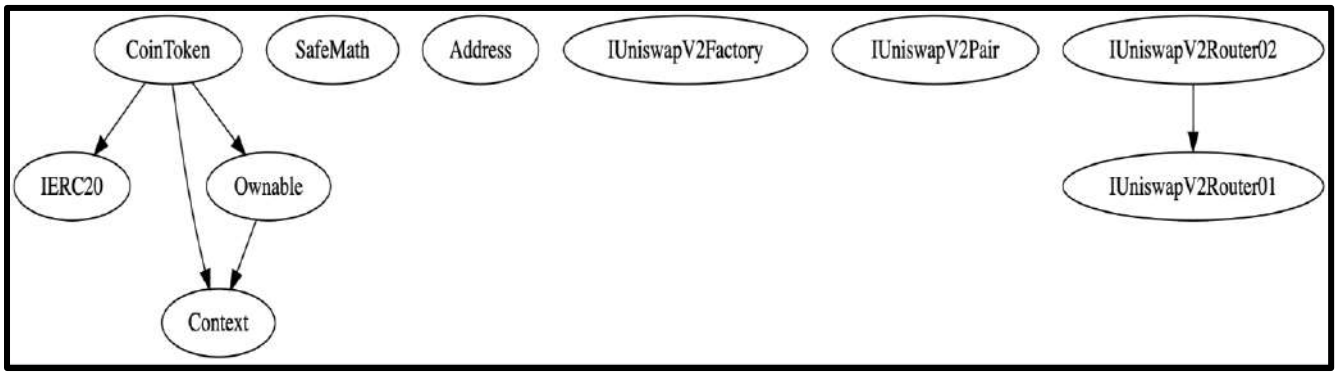
└	setTaxFeePercent	External ❗	🛑	onlyOwner
└	setDevFeePercent	External ❗	🛑	onlyOwner
└	setLiquidityFeePercent	External ❗	🛑	onlyOwner
└	setMaxTxPercent	Public ❗	🛑	onlyOwner
└	setDevWalletAddress	Public ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
└		External ❗	💵	NO❗
└	_reflectFee	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_takeLiquidity	Private 🔐	🛑	
└	_takeDev	Private 🔐	🛑	
└	calculateTaxFee	Private 🔐		
└	calculateDevFee	Private 🔐		
└	calculateLiquidityFee	Private 🔐		

└	removeAllFee	Private 🔐	🛑	
└	restoreAllFee	Private 🔐	🛑	
└	isExcludedFromFee	Public ❗		NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	_transferStandard	Private 🔐	🛑	
└	_transferToExcluded	Private 🔐	🛑	
└	_transferFromExcluded	Private 🔐	🛑	
└	setRouterAddress	External ❗	🛑	onlyOwner
└	setNumTokensSellToAddToLiquidity	External ❗	🛑	onlyOwner

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**

- **Out of gas issue.**

  In the includeInReward 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 includeInReward(address account⬆) external onlyOwner {
    require(_isExcluded[account⬆], "Account is already included");
    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 addLiquidity function calls the uniswapV2Router.addLiquidityETH function with the address specified as owner() for acquiring the generated LP tokens from the JRR Token-BNB pool. As a result, over time the _owner address will accumulate a significant portion of LP tokens. If the _owner is an EOA(Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.**

```
ftrace | funcSig
function addLiquidity(uint256 tokenAmount⬆, uint256 ethAmount⬆) private {
    _approve(address(this), address(uniswapV2Router), tokenAmount⬆);
    uniswapV2Router.addLiquidityETH{value: ethAmount⬆}(
        address(this),
        tokenAmount⬆,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```

Recommendation

We advise the address of the uniswapV2Router.addLiquidityETH function call to be replaced by the contract itself.

Owner privileges

- The owner can renounce and transfer the ownership.

```
ftrace | funcSig
function renounceOwnership() public virtual onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = address(0);
}

ftrace | funcSig
function transferOwnership(address newOwner⬆) public virtual onlyOwner {
    require(
        newOwner⬆ != address(0),
        "Ownable: new owner is the zero address"
    );
    emit OwnershipTransferred(_owner, newOwner⬆);
    _owner = newOwner⬆;
}
```

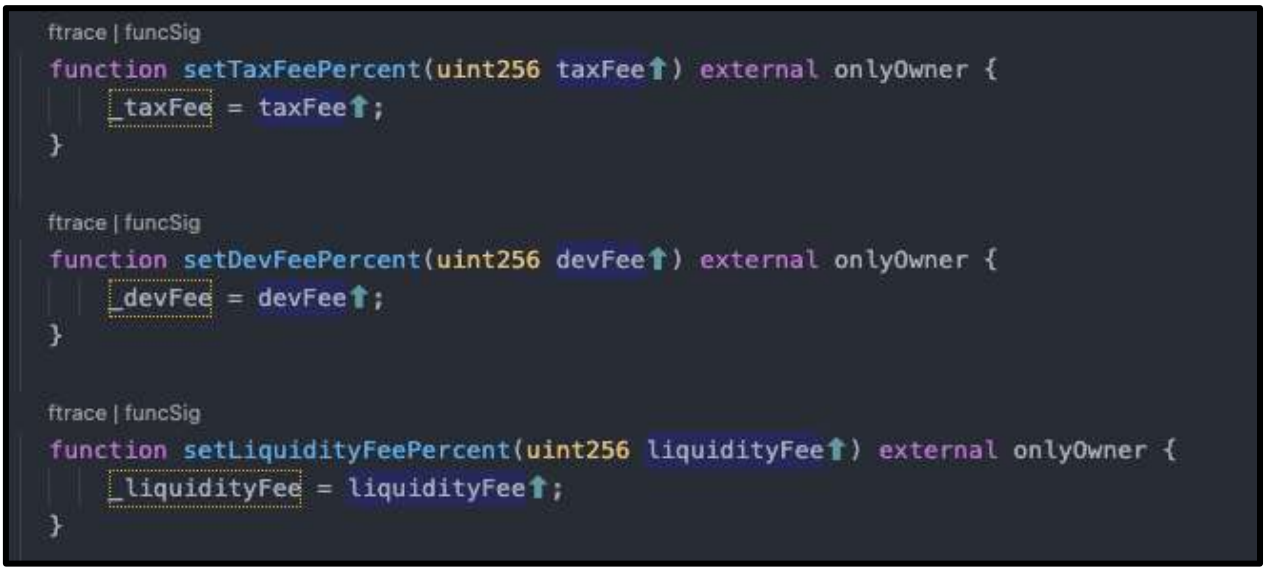
- The owner can include and exclude wallets from rewards.

```
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 included");
    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 include and exclude wallets from fee.

```
ftrace | funcSig
function excludeFromFee(address account⬆) public onlyOwner {
    _isExcludedFromFee[account⬆] = true;
}

ftrace | funcSig
function includeInFee(address account⬆) public onlyOwner {
    _isExcludedFromFee[account⬆] = false;
}
```

❖ The owner can change tax, dev and liquidity fees.



```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee⬆) external onlyOwner {
    _taxFee = taxFee⬆;
}

ftrace | funcSig
function setDevFeePercent(uint256 devFee⬆) external onlyOwner {
    _devFee = devFee⬆;
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee⬆) external onlyOwner {
    _liquidityFee = liquidityFee⬆;
}
```

❖ The owner can change the max transaction amount.

```
ftrace | funcSig
function setMaxTxPercent(uint256 maxTxPercent⬆) public onlyOwner {
    _maxTxAmount = maxTxPercent⬆ * 10**_decimals;
}
```

❖ The owner can change the dev wallet address.

```
ftrace | funcSig
function setDevWalletAddress(address _addr⬆) public onlyOwner {
    _devWalletAddress = _addr⬆;
}
```

❖ The owner can enable and disable liquidity adding.

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

❖ The owner can change the router address.

```
ftrace | funcSig
function setRouterAddress(address newRouter⬆) external onlyOwner {
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(newRouter⬆);
    uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());
    uniswapV2Router = _uniswapV2Router;
}
```

❖ The owner can change the swap point.

```
ftrace | funcSig
function setNumTokensSellToAddToLiquidity(uint256 amountToUpdate⬆)
    external
    onlyOwner
{
    numTokensSellToAddToLiquidity = amountToUpdate⬆;
}
```

Audit conclusion

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