



# **RugFreeCoins Audit**



## **BNBTiger 2.0 Token Smart Contract Security Audit**

**July 12<sup>th</sup> ,2023**



- **High severity issues**

- The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    providingLiquidity = true;
    genesis_block = block.number;
}
```

- In the code, the `_transfer` function is called before the allowance is updated. This order of operations allows a potential reentrancy attack if the recipient contract executes a malicious fallback function that calls back into the `transferFrom` function before the allowance is adjusted.

```
function transferFrom(
    address sender,
    address recipient,
    uint256 amount
) public override returns (bool) {
    _transfer(sender, recipient, amount);

    uint256 currentAllowance = _allowances[sender][_msgSender()];
    require(
        currentAllowance >= amount,
        "BEP20: transfer amount exceeds allowance"
    );
    _approve(sender, _msgSender(), currentAllowance - amount);

    return true;
}
```

# 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 enable or pause trading
- ✓ The owner can't change fees.
- ✓ 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.
- ✗ The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

# Contents

Overview	iii
Audit details	1
Disclaimer	2
Background	3
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	15
Owner privileges	17
Audit conclusion	19

# Audit details



**Audited project**  
BNBTiger 2.0 Token



**Contract Address**  
0x2622220E94E95AE0071cbdCf26A1b444d92C7dA7



**Client contact**  
BNBTiger 2.0 Token Team



**Blockchain**  
Binance Smart chain



**Project website**  
<https://bnbtiger2.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 the BNBTiger 2.0 Token Team to perform an audit of the smart contract.

<https://bscscan.com/address/0x2622220E94E95AE0071cbdCf26A1b444d92C7dA7>

This audit focuses on verifying 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.

# Tokenomics

## **2% tax when buying & selling (12/07/2023)**

- 0% of trade goes to the Liquidity pool.
- 2% of trade goes to the marketing wallet in BNB



# 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 in the BNBTiger 2.0 token ecosystem.
- Anyone who's interested in taking part in the future plans of BNBTiger 2.0 Token.
- Anyone who's interested in making financial transactions with any other party using BNBTiger 2.0 Token as the currency.

# Potential to grow with score points

1.	Project efficiency	8/10
2.	Project uniqueness	8/10
3	Information quality	8/10
4	Service quality	8/10
5	System quality	8/10
6	Impact on the community	8/10
7	Impact on the business	8/10
8	Preparing for the future	8/10
9	Smart contract security	7/10
10	Smart contract functionality assessment	9/10
Total Points		<b>8.0/10</b>

# Contract details

## Token contract details for 12<sup>th</sup> of July 2023

<b>Contract name</b>	BNBTiger2.0
<b>Contract address</b>	0x2622220E94E95AE0071cbdCf26A1b444d92C7dA7
<b>Token supply</b>	1,000,000,000,000,000
<b>Token ticker</b>	BNBTiger2.0
<b>Decimals</b>	18
<b>Token holders</b>	2
<b>Transaction count</b>	4
<b>Contract deployer address</b>	0x3Cf59A87ddE513C8BB49F5ecF70757C083B6aaaa
<b>Contract's current owner address</b>	0x484cc7fcdb22ee5751252c62e74615d656f8cfab
<b>Marketing wallet</b>	0xc12445fe8b4e0e4c31b7f61308867f508672a9a5

# 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 ISSUES
		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 & centralization	Access control of owners	pass
		Business logics	pass
		Business implementations	pass
4	Integer overflow/underflow		pass
5	Reentrancy		HIGH ISSUES
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
Context	Implementation			
L	_msgSender	Internal 		
L	_msgData	Internal 		
IBEP20	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 !
IBEP20Meta data	Interface	IBEP20		
L	name	External !		NO !
L	symbol	External !		NO !
L	decimals	External !		NO !

BEP20	Implementation	Context, IBEP20, IBEP20 Metadata		
L		Public !		NO !
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	_transfer	Internal 		
L	_tokengeneration	Internal 		
L	_approve	Internal 		
Address	Library			
L	sendValue	Internal 		

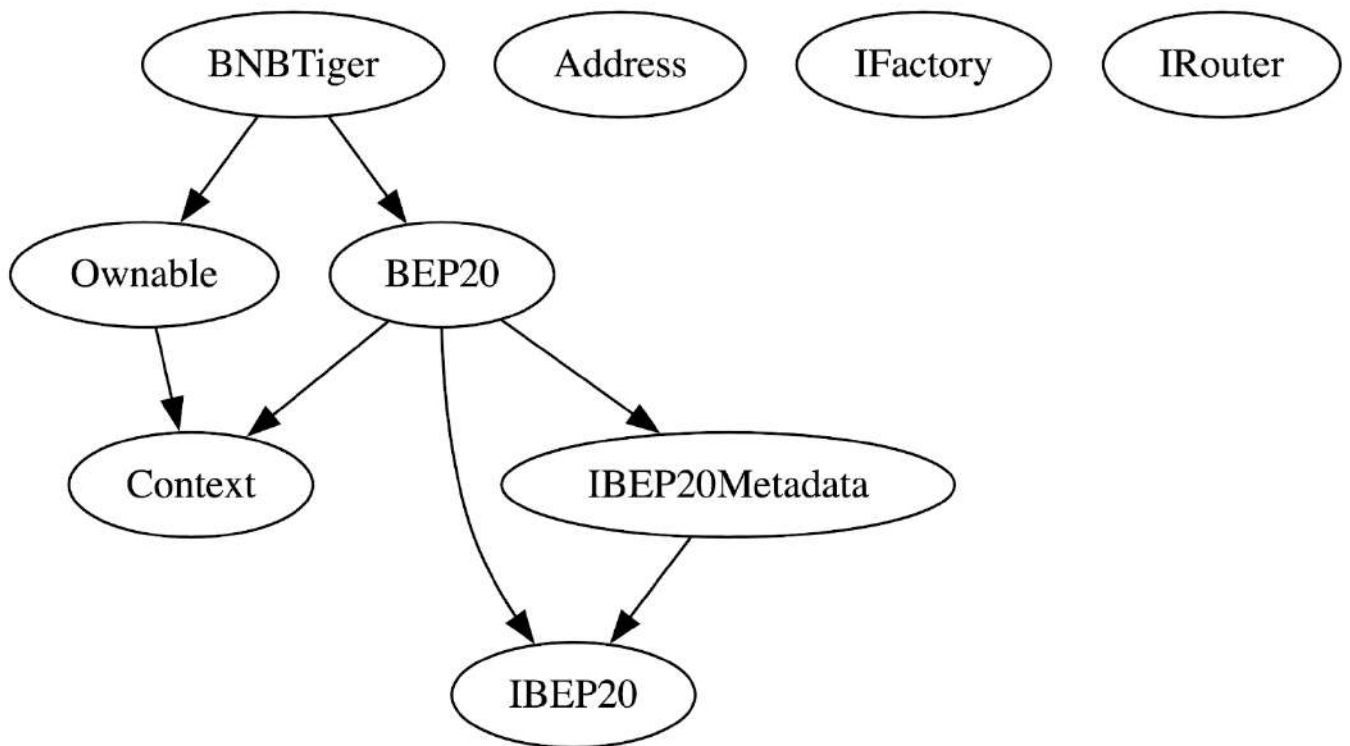
<b>Ownable</b>	<b>Implementation</b>	<b>Context</b>		
L		Public !		NO !
L	owner	Public !		NO !
L	renounceOwnership	Public !		onlyOwner
L	transferOwnership	Public !		onlyOwner
L	_setOwner	Private 🗝️		
<b>IFactory</b>	<b>Interface</b>			
L	createPair	External !		NO !
<b>IRouter</b>	<b>Interface</b>			
L	factory	External !		NO !
L	WETH	External !		NO !
L	addLiquidityETH	External !		NO !
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !		NO !
<b>BNBTiger</b>	<b>Implementation</b>	<b>BEP20, Ownable</b>		
L		Public !		BEP20
L	approve	Public !		NO !
L	transferFrom	Public !		NO !
L	increaseAllowance	Public !		NO !
L	decreaseAllowance	Public !		NO !

L	transfer	Public !		NO !
L	_transfer	Internal 		
L	Liquify	Private 		lockThe Swap
L	swapTokensForETH	Private 		
L	addLiquidity	Private 		
L	updateLiquidityProvide	External !		onlyOwner
L	updateLiquidityTreshhold	External !		onlyOwner
L	EnableTrading	External !		onlyOwner
L	updatedeadline	External !		onlyOwner
L	updateMarketingWallet	External !		onlyOwner
L	updateExemptFee	External !		onlyOwner
L	bulkExemptFee	External !		onlyOwner
L	rescueBNB	External !		onlyOwner
L	rescueBSC20	External !		onlyOwner
L		External !		NO !

### Legend

Symbol	Meaning
	Function can modify state
	Function is payable

## Inheritance Hierarchy





# Security issue checking status

## ❖ High severity issues

In the code, the `_transfer` function is called before the allowance is updated. This order of operations allows a potential reentrancy attack if the recipient contract executes a malicious fallback function that calls back into the `transferFrom` function before the allowance is adjusted.

```
function transferFrom(
    address sender,
    address recipient,
    uint256 amount
) public override returns (bool) {
    _transfer(sender, recipient, amount);

    uint256 currentAllowance = _allowances[sender][_msgSender()];
    require(
        currentAllowance >= amount,
        "BEP20: transfer amount exceeds allowance"
    );
    _approve(sender, _msgSender(), currentAllowance - amount);

    return true;
}
```

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    providingLiquidity = true;
    genesis_block = block.number;
}
```

## ❖ Medium severity issues

No medium severity issues found

### ❖ Low severity issues

feeSum and feeswap variables always have the same value, so having two different variables is not needed, and can remove one to save gas.

```
uint256 feeswap;  
uint256 feesum;  
uint256 fee;  
Taxes memory currentTaxes;  
  
bool useLaunchFee = !exemptFee[sender] &&  
    !exemptFee[recipient] &&  
    block.number < genesis_block + deadline;  
  
//set fee to zero if fees in contract are handled or exempted  
if (!_interlock || exemptFee[sender] || exemptFee[recipient])  
    fee = 0;  
  
    //calculate fee  
else if (recipient == pair && !useLaunchFee) {  
    feeswap = sellTaxes.liquidity + sellTaxes.marketing;  
    feesum = feeswap;  
    currentTaxes = sellTaxes;  
} else if (!useLaunchFee) {  
    feeswap = taxes.liquidity + taxes.marketing;  
    feesum = feeswap;  
    currentTaxes = taxes;  
} else if (useLaunchFee) {  
    feeswap = launchtax;  
    feesum = launchtax;  
}  
  
fee = (amount * feesum) / 100;
```

### ❖ Centralization Risk

No Centralization issues found

# Owner privileges

- ❖ The owner can enable/disable adding lp

```
function updateLiquidityProvide(bool state) external onlyOwner {  
    providingLiquidity = state;  
}
```

- ❖ The owner can change the swap threshold maximum upto 50% and minimum no limit

```
function updateLiquidityTreshhold(uint256 new_amount) external onlyOwner {  
    require(  
        new_amount <= 5e14,  
        "Swap threshold amount should be lower or equal to 50% of tokens"  
    );  
    tokenLiquidityThreshold = new_amount * 10 ** decimals();  
}
```

- ❖ The owner can enable trading, but once enabled can not disable it again

```
function EnableTrading() external onlyOwner {  
    require(!tradingEnabled, "Cannot re-enable trading");  
    tradingEnabled = true;  
    providingLiquidity = true;  
    genesis_block = block.number;  
}
```

- ❖ The owner can add dead blocks a maximum up to 4 before enabling trading

```
function updatedeadline(uint256 _deadline) external onlyOwner {  
    require(!tradingEnabled, "Can't change when trading has started");  
    require(_deadline < 5, "Deadline should be less than 5 Blocks");  
    deadline = _deadline;  
}
```

- ❖ The owner can change marketing wallet address

```
function updateMarketingWallet(address newWallet) external onlyOwner {
    require(newWallet != address(0), "Fee Address cannot be zero address");
    marketingWallet = newWallet;
}
```

- ❖ The owner can include/exclude wallets from fees

```
function updateExemptFee(address _address, bool state) external onlyOwner {
    exemptFee[_address] = state;
}

function bulkExemptFee(
    address[] memory accounts,
    bool state
) external onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
        exemptFee[accounts[i]] = state;
    }
}
```

- ❖ The owner can get contract BNB to owner wallet

```
function rescueBNB(uint256 weiAmount) external onlyOwner {
    payable(owner()).transfer(weiAmount);
}
```

- ❖ The owner can get any bep20 tokens from the contract but can get native tokens

```
function rescueBSC20(address tokenAdd, uint256 amount) external onlyOwner {
    require(
        tokenAdd != address(this),
        "Owner can't claim contract's balance of its own tokens"
    );
    IBEP20(tokenAdd).transfer(owner(), amount);
}
```

# 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: **3**

Smart contract security: **HIGH RISK**

Solidity code functional issue level: **PASS**

Number of owner privileges: **8**

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

Smart contract active ownership: **ACTIVE**