



# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

```
// ---- \\  
// ///..) \\  
|| <|| >\\ ||  
|| $$ --/ ||  
*\\ |\\_ / // *  
*\\/_---\\_// *  
/ BRRRRR \\
```

**Brr**

\$Brr

07/07/2023



# TOKEN OVERVIEW

---

## Fees

- Buy fees: 3%
- Sell fees: 3%

## Fees privileges

- Can change buy fees up to 25% and sell fees up to 25%

## Ownership

- Owned

## Minting

- No mint function

## Max Tx Amount / Max Wallet Amount

- Can change max tx amount and max wallet amount (with threshold)

## Blacklist

- Blacklist function not detected

## Other privileges

- Can exclude / include from fees
  - Can exclude / include from tx limitations
-

# TABLE OF CONTENTS

1

DISCLAIMER

2

INTRODUCTION

3

WEBSITE + SOCIALS

4-5

AUDIT OVERVIEW

6-9

OWNER PRIVILEGES

10

CONCLUSION AND ANALYSIS

11

TOKEN DETAILS

12

BRR TOKEN ANALYTICS &  
TOP 10 TOKEN HOLDERS

13

TECHNICAL DISCLAIMER



# DISCLAIMER

The information provided on this analysis document is only for general information and should not be used as a reason to invest.

FreshCoins Team will take no payment for manipulating the results of this audit.

The score and the result will stay on this project page information on our website <https://freshcoins.io>

FreshCoins Team does not guarantees that a project will not sell off team supply, or any other scam strategy ( RUG or Honeypot etc )



# INTRODUCTION

**FreshCoins** (Consultant) was contracted by **Brr** (Customer) to conduct a Smart Contract Code Review and Security Analysis.

**0x31e35EeC73f8FAc94bB339295B95f842CB7fF26d**

Network: **Ethereum (ETH)**

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on **07/07/2023**



# WEBSITE DIAGNOSTIC

<https://www.brr-token.xyz/>



# 0-49



## 50-89



## 90-100



## Performance



## Accessibility



## Best Practices



## SEO



# Progressive Web App

## Socials



Twitter

<https://twitter.com/BrrToken>



Telegram

<https://t.me/BRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR>

# AUDIT OVERVIEW



Security Score



## Static Scan

Automatic scanning for common vulnerabilities



## ERC Scan

Automatic checks for ERC's conformance



High



Medium



Low



Optimizations



Informational



No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Passed
3	Front running	Passed
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Passed
7	DoS with block gas limit	Passed
8	Methods execution permissions	Passed
9	Exchange rate impact	Passed
10	Malicious Event	Passed
11	Scoping and Declarations	Passed
12	Uninitialized storage pointers	Passed
13	Design Logic	Passed
14	Safe Zeppelin module	Passed



# OWNER PRIVILEGES

- Contract owner can't mint tokens after initial contract deploy
- Contract owner can't exclude an address from transactions
- Contract owner can exclude/include wallet from tax

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
    _isExcludedFromFees[account] = excluded;
    emit ExcludeFromFees(account, excluded);
}
```

- Contract owner can exclude/include wallet from tx limitations

```
function excludeFromMaxTransaction(address updAds, bool isEx) public onlyOwner {
    _isExcludedMaxTransactionAmount[updAds] = isEx;
}
```

- Contract owner can change swap settings

```
function updateSwapEnabled(bool enabled) external onlyOwner {
    swapEnabled = enabled;
}

function updateSwapTokensAtAmount(
    uint256 newAmount
) external onlyOwner returns (bool) {
    require(
        newAmount >= totalSupply()/100000,
        "Must be greater than 0.001%"
    );
    swapTokensAtAmount = newAmount;
    return true;
}
```

- Contract owner can change tx limitations and wallet limitations  
(with threshold)

```
function updateMaxTxnAmount(uint256 newNum) external onlyOwner {
    require(newNum >= ((totalSupply() * 5) / 1000) / 1e18, "Cannot set max wallet amount lower than 0.5%");
    require(newNum <= ((totalSupply() * 5) / 100) / 1e18, "Cannot set max wallet amount higher than 5%");
    maxTransactionAmount = newNum * (10 ** 18);
}

function updateMaxWalletAmount(uint256 newNum) external onlyOwner {
    require(newNum >= ((totalSupply() * 5) / 1000) / 1e18, "Cannot set max wallet amount lower than 0.5%");
    require(newNum <= ((totalSupply() * 5) / 100) / 1e18, "Cannot set max wallet amount higher than 5%");
    maxWallet = newNum * (10 ** 18);
}
```

## ● Contract owner can remove all limitations (once disabled, can't be enabled)

```
function removeLimits() external onlyOwner returns (bool) {
    limitsInEffect = false;
    return true;
}
```

## ● Contract owner can change buy fees up to 25% and sell fees up to 25%

```
function updateBuyFees(
    uint256 _marketingFee,
    uint256 _devFee,
    uint256 _buyTreasuryFee
) external onlyOwner {
    buyMarketingFee = _marketingFee;
    buyDevFee = _devFee;
    buyTreasuryFee = _buyTreasuryFee;
    buyTotalFees = buyMarketingFee + buyDevFee + buyTreasuryFee;
    require(buyTotalFees <= 25, "");
}

function updateSellFees(
    uint256 _marketingFee,
    uint256 _devFee,
    uint256 _buyTreasuryFee
) external onlyOwner {
    sellMarketingFee = _marketingFee;
    sellDevFee = _devFee;
    sellTreasuryFee = _buyTreasuryFee;
    sellTotalFees = sellMarketingFee + sellTreasuryFee + sellDevFee;
    require(sellTotalFees <= 25, "");
}
```

## ● Contract owner can change marketingWallet, treasuryWallet and devWallet addresses

Current values:

**marketingWallet:** 0x11E33Ca6042344B274e03D60077Bc809bAbd318E

**treasuryWallet:** 0x6bC8F8325d42497b9bE8c376F56DDF0731BBA775

**devWallet:** 0x11E33Ca6042344B274e03D60077Bc809bAbd318E

```
function updateMarketingWallet(
    address newMarketingWallet
) external onlyOwner {
    emit marketingWalletUpdated(newMarketingWallet, marketingWallet);
    marketingWallet = newMarketingWallet;
}

function updateTreasuryWallet(address newWallet) external onlyOwner {
    emit treasuryWalletUpdated(newWallet, treasuryWallet);
    treasuryWallet = newWallet;
}

function updateDevWallet(address newWallet) external onlyOwner {
    emit devWalletUpdated(newWallet, devWallet);
    devWallet = newWallet;
}
```

- **Contract owner has ability to retrieve any token held by the contract**  
(native tokens included)

```
function withdrawETH(uint256 _amount) external onlyOwner {
    require(address(this).balance >= _amount, "Invalid Amount");
    payable(msg.sender).transfer(_amount);
}

function withdrawToken(IERC20 _token, uint256 _amount) external onlyOwner {
    require(_token.balanceOf(address(this)) >= _amount, "Invalid Amount");
    _token.transfer(msg.sender, _amount);
}
```

- **Contract owner has the ability to send tokens to multiple wallets using a single transaction**

If the airdrop operation involves a large number of recipients, it may consume a significant amount of gas, potentially leading to out-of-gas errors

```
function airdrop(
    address[] calldata addresses,
    uint256[] calldata amounts
) external onlyOwner {
    require(
        addresses.length == amounts.length,
        "Array sizes must be equal"
    );
    uint256 i = 0;
    while (i < addresses.length) {
        uint256 _amount = amounts[i].mul(1e18);
        _transfer(msg.sender, addresses[i], _amount);
        i += 1;
    }
}
```

- **Contract owner can transfer ownership**

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(
        newOwner != address(0),
        "Ownable: new owner is the zero address"
    );
    _transferOwnership(newOwner);
}

function _transferOwnership(address newOwner) internal virtual {
    address oldOwner = _owner;
    _owner = newOwner;
    emit OwnershipTransferred(oldOwner, newOwner);
}
```

- **Contract owner can renounce ownership**

```
function renounceOwnership() public virtual onlyOwner {
    _transferOwnership(address(0));
}
```

### **Recommendation:**

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. The risk can be prevented by temporarily locking the contract or renouncing ownership.



# CONCLUSION AND ANALYSIS



Smart Contracts within the scope were manually reviewed and analyzed with static tools.



Audit report overview contains all found security vulnerabilities and other issues in the reviewed code.



Found no HIGH issues during the first review.

# TOKEN DETAILS

## Details

Buy fees:	3%
Sell fees:	3%
Max TX:	10,000,000,000
Max Sell:	N/A

## Honeypot Risk

Ownership:	Owned
Blacklist:	Not detected
Modify Max TX:	Detected
Modify Max Sell:	Not detected
Disable Trading:	Not detected

## Rug Pull Risk

Liquidity:	N/A
Holders:	19% unlocked tokens



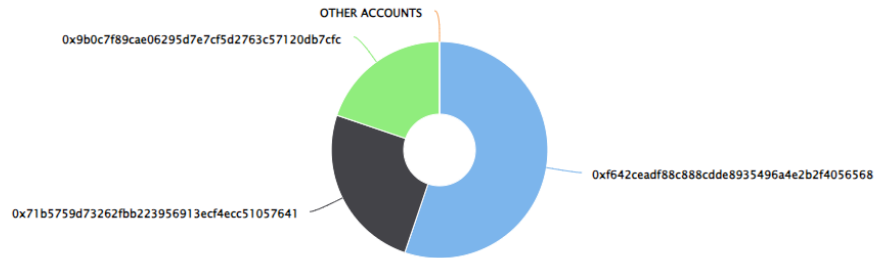
# BRR TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (1,000,000,000,000.00 Tokens) of Brr

Token Total Supply: 1,000,000,000,000.00 Token | Total Token Holders: 3

Brr Top 10 Token Holders

Source: Etherscan.io



(A total of 1,000,000,000,000.00 tokens held by the top 10 accounts from the total supply of 1,000,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	<a href="#">0xF642cE...f4056568</a>	552,000,000,000	55.2000%
2	<a href="#">0x71B575...51057641</a>	250,000,000,000	25.0000%
3	<a href="#">0x9B0c7f...20Db7Cfc</a>	198,000,000,000	19.8000%

# TECHNICAL DISCLAIMER

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. The audit can't guarantee the explicit security of the audited project / smart contract.

