freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Dogun

$DOGUN

07/01/2025

TOKEN OVERVIEW

Fees

- Buy fees: 5% (check page 7)
- Sell fees: 5% (check page 7)

Fees privileges

- Can change buy fees up to 5% and sell fees up to 5% (check page 7)

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

- Can't change max tx amount and / or max wallet amount

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude/include wallet from fees
- Contract owner has to call startTrading function to enable trade

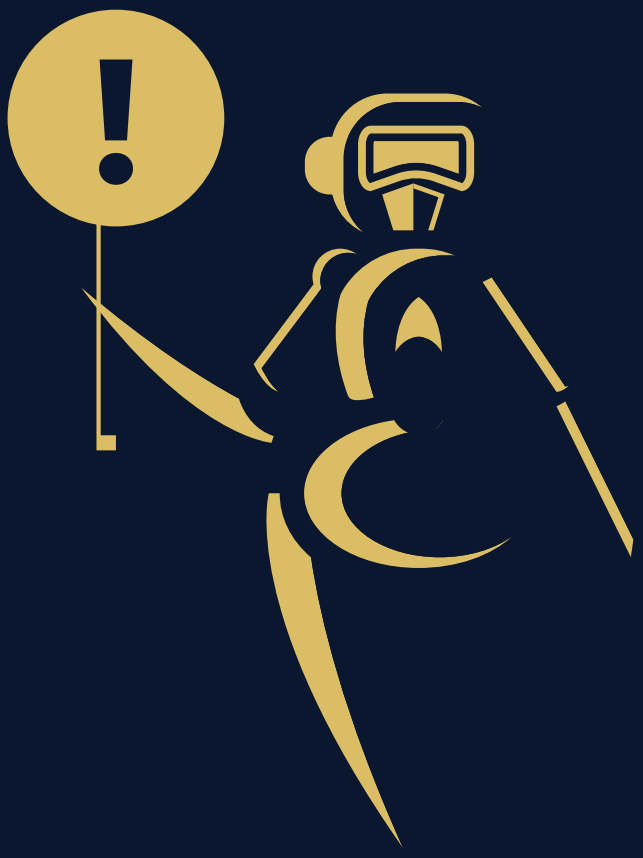
TABLE OF CONTENTS

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 Dogun (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0xBa7e0D4Eb6b92692b2641dCe9c22aE5a64b336E0

Network: Binance Smart Chain (BSC)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 07/01/2025

WEBSITE DIAGNOSTIC

https://www.dogun.io


0-49


50-89


90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials


X (Twitter)

https://x.com/dogunbsc


Telegram

https://t.me/DogunBsc

AUDIT OVERVIEW

82

Security Score

88

Static Scan
Automatic scanning for common vulnerabilities

80

ERC Scan
Automatic checks for ERC's conformance

1 High

1 Medium

0 Low

0 Optimizations

0 Informational

No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Low
3	Front running	Low
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 setisExempt(address _address, bool _enabled) external onlyOwner {
    isFeeExempt[_address] = _enabled;
}
```

- Contract owner has to call startTrading function to enable trade

Please note that any wallet excluded from fees retains the ability to engage in trading, even in situations where trading has been disabled

```
function startTrading() external onlyOwner {
    require(!tradingOpen, "Trading already enabled");
    tradingOpen = true;
    initStartTime = block.timestamp; // Set the start time to the current block timestamp
}

.
.
.
function checkTradingAllowed(
    address sender,
    address recipient
) internal view {
    if (!isFeeExempt[sender] && !isFeeExempt[recipient]) {
        require(tradingOpen, "ERC20: Trading is not allowed");
    }
}
.
```

- Contract owner has ability to retrieve any token held by the contract

Native tokens excluded

```
function rescueBNB() external onlyMarketing {
    payable(msg.sender).transfer(address(this).balance);
}

function rescueTokens(address _address, uint256 _amount) external onlyMarketing {
    require(_address != address(this), "Cannot rescue this token itself");
    IERC20(_address).transfer(msg.sender, _amount);
}
```

Contract owner can change marketing_receiver and dev_receiver addresses

Current values:

marketing_receiver: 0x538455683e2A18740417891D4486820526dC5e7a

dev_receiver: 0x585D1508954890AB0F72750Dc2fD43da19E3E596

```
function setMarketingReceiver(address newReceiver) external onlyOwner {
    require(newReceiver != address(0), "Invalid address: zero address");
    marketing_receiver = payable(newReceiver);
    emit MarketingReceiverUpdated(newReceiver);
}

function setDevReceiver(address newReceiver) external onlyOwner {
    require(newReceiver != address(0), "Invalid address: zero address");
    dev_receiver = payable(newReceiver);
    emit DevReceiverUpdated(newReceiver);
}
```

Dynamic Fees:

Initial High Fees: The contract starts with a high tax rate (30%) for the first few minutes after trading opens:

BUY_TAX_DURATION: During the first 3 minutes, a buy fee of up to 30% is applied.

SELL_TAX_DURATION: During the first 30 minutes, a sell fee of up to 30% is applied.

Over time, these initial fees reduce linearly to the regular buy and sell fee levels (buyFee and sellFee). Regular buy and sell fees are capped at 5% each (MAX_BUY_FEE and MAX_SELL_FEE).

```
function getTotalFee(
    address sender,
    address recipient
  ) public view returns (uint256) {
    if (recipient == pair) {
      // Selling
      if (block.timestamp < initStartTime + SELL_TAX_DURATION) {
        uint256 timeElapsed = block.timestamp - initStartTime;
        uint256 sellTax = INITIAL_SELL_TAX -
          ((INITIAL_SELL_TAX - sellFee) * timeElapsed) /
          SELL_TAX_DURATION;
        return sellTax;
      }
      return sellFee; // After duration, return sellFee
    }
    if (sender == pair) {
      // Buying
      if (block.timestamp < initStartTime + BUY_TAX_DURATION) {
        uint256 timeElapsed = block.timestamp - initStartTime;
        uint256 buyTax = INITIAL_BUY_TAX -
          ((INITIAL_BUY_TAX - buyFee) * timeElapsed) /
          BUY_TAX_DURATION;
        return buyTax;
      }
      return buyFee; // After duration, return buyFee
    }
    return 0; // For transfers
}
```

Contract owner can change swap settings

```
function setSwapAndLiqEnabled(bool _enabled) external onlyOwner {
    swapAndLiqEnabled = _enabled;
    emit SwapAndLiqEnabledUpdated(_enabled);
}
```

Contract owner can exclude/include wallet from tax

```
uint256 private constant MAX_BUY_FEE = 500; // 5%
uint256 private constant MAX_SELL_FEE = 500; // 5%
```

```
function setStructure(uint256 _buy, uint256 _sell) external onlyOwner {
    require(
      _buy <= MAX_BUY_FEE && _sell <= MAX_SELL_FEE,
      "Fees exceed maximum limits"
    );
    buyFee = _buy;
    sellFee = _sell;
    emit StructureUpdated(_buy, _sell);
}
```

Contract owner can transfer ownership

```
function transferOwnership(address payable newOwner) public onlyOwner {
        require(
          newOwner != address(0),
          "Ownable: new owner is the zero address"
        );
        emit OwnershipTransferred(owner, newOwner); // Emit old and new owner
        owner = newOwner;
    }
```

Contract owner can renounce ownership

```
function renounceOwnership() public onlyOwner {
    emit OwnershipTransferred(owner, address(0));
    owner = 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 1 HIGH issue during the first review.

TOKEN DETAILS

Details

Buy fees:	5% (check page 7)
Sell fees:	5% (check page 7)
Max TX:	N/A
Max Sell:	N/A

Honeypot Risk

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

Rug Pull Risk

Liquidity:	N/A
Holders:	100% unlocked tokens

DOGUN TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (100,000,000,000.00 Tokens) of Dogun

Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 1

(A total of 100,000,000,000.00 tokens held by the top 10 accounts from the total supply of 100,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	Kitty Run: Deployer	100,000,000,000	100.0000%

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.