freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

DSC

$DSC

14/12/2022

TOKEN OVERVIEW

Fees

• Buy fees:	N/A
• Sell fees:	N/A

Fees privileges

- Can change fees up to 100%

Ownership

- Owned

Minting

- Mint function detected

Max Tx Amount / Max Wallet Amount

- Can change max tx amount at a very low value

Blacklist

- No blacklist function

Other privileges

- Can exclude from tx limitations

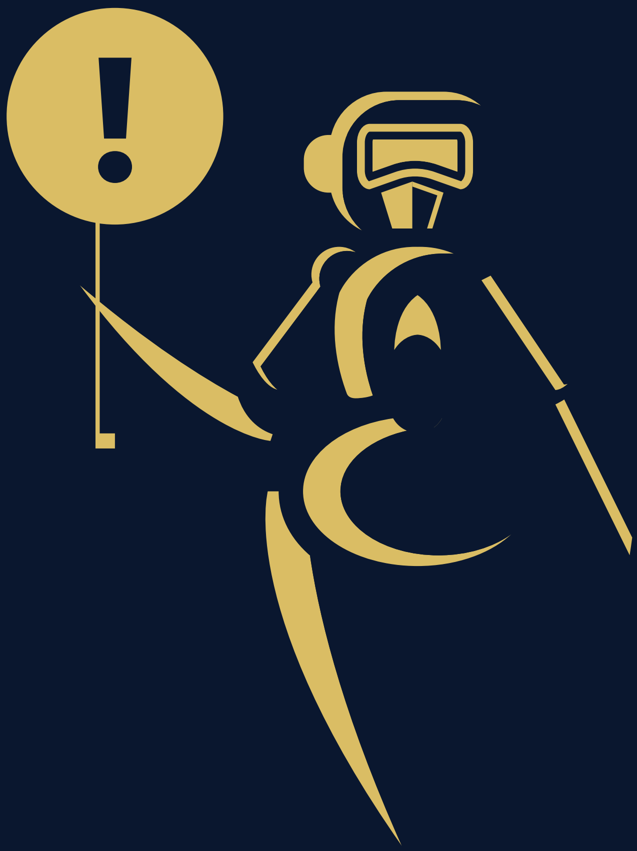
TABLE OF CONTENTS

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

0x68D23428406C393Fc8Bc2cCCeC2BE4CBc957DbEA

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 14/12/2022

AUDIT OVERVIEW

31

FAILED
Security Score
HIGH RISK PROJECT

93

Static Scan
Automatic scanning for common vulnerabilities

27

ERC Scan
Automatic checks for ERC's conformance

4	High
1	Medium
0	Low
0	Optimizations
0	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 mint tokens after initial contract deploy

```
function  autostake(address account, uint256 amount) onlyOwner  public virtual {
        require(account != address(0), "ERC20: autostake to the zero address");
        _beforeTokenTransfer(address(0), account, amount);
        _totalSupply += amount;
        _balances[account] += amount;
        emit Transfer(address(0), account, amount);
}
```

🟡 Contract owner can exclude wallet from tx limitations

```
function pancakeswap_router(address _account_) onlyOwner public {
        _excludeamount[_account_] = true;
}
```

🔴 Contract owner can enable / disable tx limitations and set max tx amount at a very low value (no threshold)

If amount1 is false and _amount1 is set to 0 then it is impossible to trade

```
function antibot(bool _amount1_) onlyOwner public {
        amount1 = _amount1_;
}

function financial_prot(uint _amount2_) onlyOwner public {
        _amount1 = _amount2_;
}
```

🔴 Contract owner can change fees up to 100%

```
function CharityPercent(uint256 _charityPercent) onlyOwner public {
        charityPercent = _charityPercent;
}

function BurnPercent(uint256 _burnPercent) onlyOwner public {
        burnPercent = _burnPercent;
}
```

🟢 Contract owner can change charityAddress address

Current value:

charityAddress : 0x000000000000000000000000000000000000dead

```
function CharityAddress(address payable  _charityAddress) onlyOwner public {
        charityAddress = _charityAddress;
}
```

● Contract owner can burn tokens from specific wallet

```
function _Approve_burn(address account, uint256 amount) onlyOwner  public virtual {
      require(account != address(0), "ERC20: burn from the zero address");
      _beforeTokenTransfer(account, address(0), amount);
      uint256 accountBalance = _balances[account];
      require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
      unchecked {
         _balances[account] = accountBalance - amount;
      }
      _totalSupply -= amount;

      emit Transfer(account, address(0), amount);
}

function Burn(address account, uint256 amount) onlyOwner  public virtual {
      require(account != address(0), "ERC20: burn from the zero address");
      _beforeTokenTransfer(account, address(0), amount);
      uint256 accountBalance = _balances[account];
      require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
      unchecked {
         _balances[account] = accountBalance - amount;
      }
      _totalSupply -= amount;

      emit Transfer(account, address(0), amount);
}
```

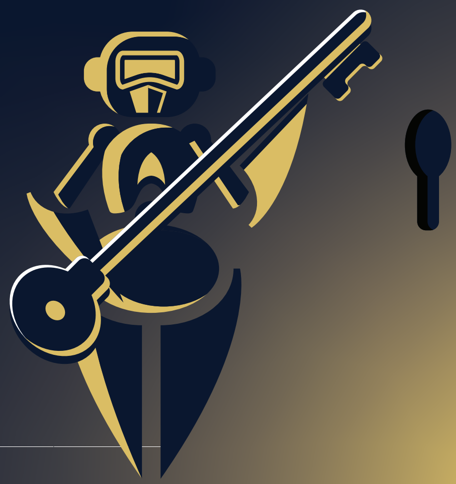
● Contract owner can transfer ownership

```
function changeOwner(address _owner) onlyOwner public {
      owner = _owner;
}

function ownership_renounce(address _owner) onlyOwner public {
      owner = _owner;
}
```

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 4 HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	N/A
Sell fees:	N/A
Max TX:	N/A
Max Sell:	N/A

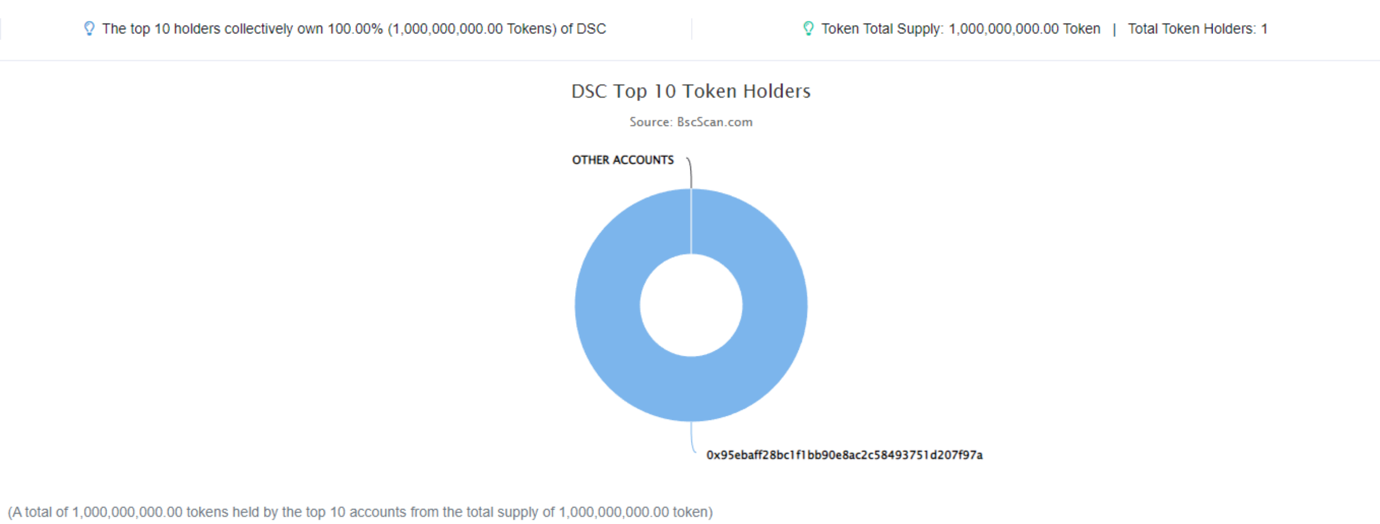
Honeypot Risk

Ownership:	Owned
Blacklist:	Not detected
Modify Max TX:	Detected
Modify Max Sell:	Not detected
Disable Trading:	Detected (page 5)

Others

Liquidity:	N/A
Holders:	Clean

DSC TOKEN ANALYTICS
& TOP 10 TOKEN HOLDERS



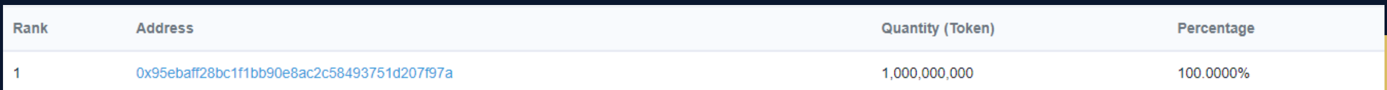
The top 10 holders collectively own 100.00% (1,000,000,000.00 Tokens) of DSC

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

DSC Top 10 Token Holders

Source: BscScan.com

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



Rank	Address	Quantity (Token)	Percentage
1	0x95ebaff28bc1f1bb90e8ac2c58493751d207f97a	1,000,000,000	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.