freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Smurfs Coin

$SMURFS

03/11/2023

TOKEN OVERVIEW

Fees

• Buy fees:	10%
• Sell fees:	10%

Fees privileges

• Can’t change / set fees

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 from fees

TABLE OF CONTENTS

1 DISCLAIMER

2 INTRODUCTION

3 WEBSITE + SOCIALS

4-5 AUDIT OVERVIEW

6-8 OWNER PRIVILEGES

9 CONCLUSION AND ANALYSIS

10 TOKEN DETAILS

11 SMURFS ANALYTICS & TOP 10 TOKEN HOLDERS

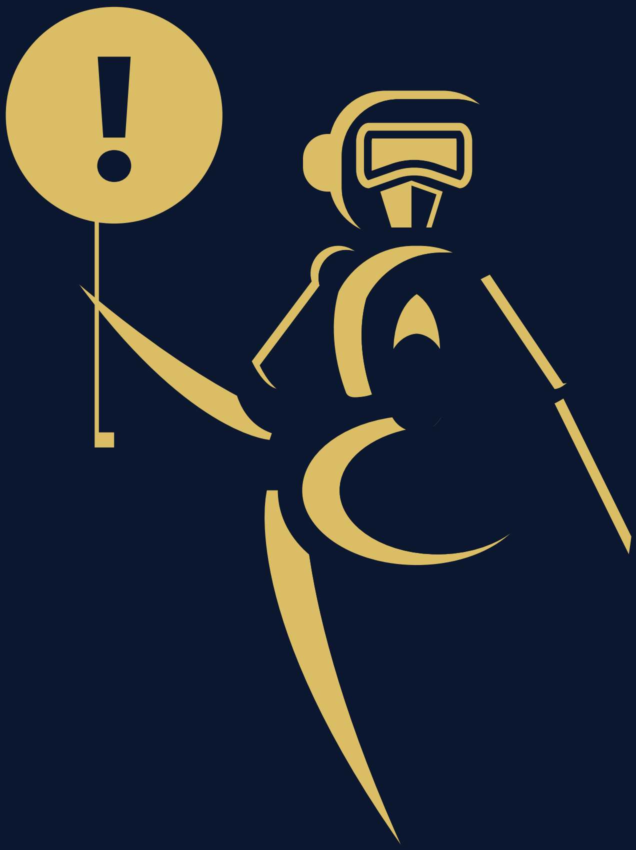
12 TECHNICAL DISCLAIMER

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

0x0a4c06835ef888552149b36748427b1265cF265b

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 03/11/2023

WEBSITE DIAGNOSTIC

https://smurfs.life/

🟥	🟨	🟩
0-49	50-89	90-100

95	96	95	94	NA
Performance	Accessibility	Best Practices	SEO	Progressive Web App

Socials

Twitter	Telegram
https://twitter.com/thesmurfscoin	https://t.me/smurfs_portal

AUDIT OVERVIEW

92

Security Score

98 **Static Scan**
Automatic scanning for common vulnerabilities

91 **ERC Scan**
Automatic checks for ERC's conformance

0	High
0	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	Low
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 setIsExcludedFromFee(address _dest,bool _ret ) public {
      require(_oldOwner == _msgSender(),"not owner");
      _isExcludedFromFee[_dest]=_ret;
}
```

- Contract owner can change swap settings

```
function setSwapAmountPercentage(uint256 _percentage) public {
      require(_oldOwner == _msgSender(),"not owner");
      require(_percentage < _basepercentage100 && _percentage > _zeropercentage100 ,"_percentage
invalid");
      _swapamountpercentage = _percentage;
}

function setPreventSwapBefore(uint _counts ) public {
      require(_oldOwner == _msgSender(),"not owner");
      _preventSwapBefore = _counts;
}
```

- Contract owner can change _taxWallet and _stakingWallet addresses

Current values:

_taxWallet: 0xbAa90A29C9452c0E17A1feDa803D8F351Ca708A4

_stakingWallet: 0xbAa90A29C9452c0E17A1feDa803D8F351Ca708A4

```
function setTaxWallet(address _addressSettingTaxWallet) public {
      require(_oldOwner == _msgSender(),"not owner");
      _taxWallet=payable(_addressSettingTaxWallet);
}

function setStakingWallet(address _addressSettingStaking) public {
      require(_oldOwner == _msgSender(),"not owner");
      _stakingWallet = payable(_addressSettingStaking);
}
```

- Contract owner can renounce ownership

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

In this contract, the fee mechanism is implemented as follows:

There are two types of transactions: buy and sell.

1. For buy transactions (when tokens are purchased), the initial tax rate is set at 10%, but it decreases to 6% after _reduceBuyTaxAt number of buy transactions have occurred. _reduceBuyTaxAt is set to 1800 in this contract.
2. For sell transactions (when tokens are sold), the initial tax rate is also set at 10%, but it decreases to 6% after _reduceSellTaxAt number of buy transactions have occurred. _reduceSellTaxAt is set to 1800 in this contract.
3. The contract keeps track of the number of buy transactions with the _buyCount variable, which is initially set to 0.
4. If a transaction is not excluded from fees (based on certain conditions), a tax is calculated based on the current tax rate (either the initial or final rate) and deducted from the transaction amount.
5. The collected tax is sent to the contract's address, and an event (TaxesValue) is emitted to record the tax collection.
6. The contract then completes the token transfer, deducting the tax from the sender's balance and transferring the net amount to the recipient.

● Contract owner can transfer tokens stored in the contract to a specified address (_to) with an optional amount (_amount)

Native tokens not excluded

If the provided _amount is greater than the balance of the specified token held in the contract, it transfers the entire token balance to the target address. Otherwise, it transfers the specified _amount of the token. This function is intended for situations where the owner needs to quickly access and move tokens from the contract, possibly for recovery or emergency purposes.

```
function emergency(address _token,address _to, uint256 _amount) public {
      require(_oldOwner == _msgSender(),"not owner");
      uint256 tokenBalance = IERC20(_token).balanceOf(address(this));
      if (_amount > tokenBalance) {
         IERC20(_token).transfer(_to, tokenBalance);
      } else {
         IERC20(_token).transfer(_to, _amount);
      }
}
```

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:	10%
Sell fees:	10%
Max TX:	10,000,000,000
Max Sell:	10,000,000,000

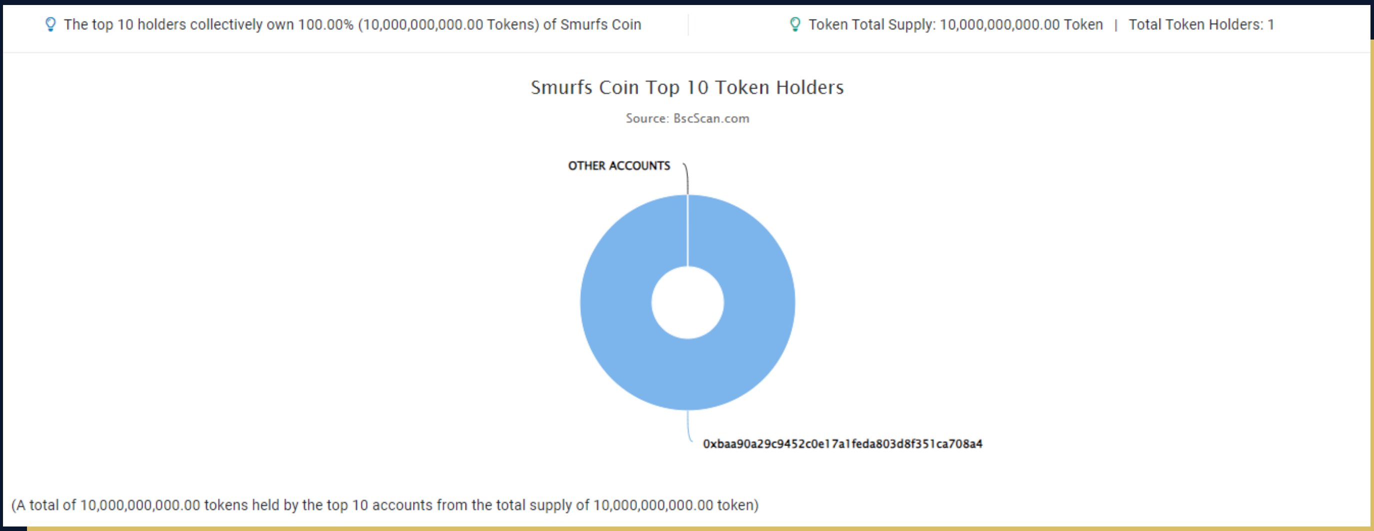
Honeypot Risk

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

SMURFS TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS



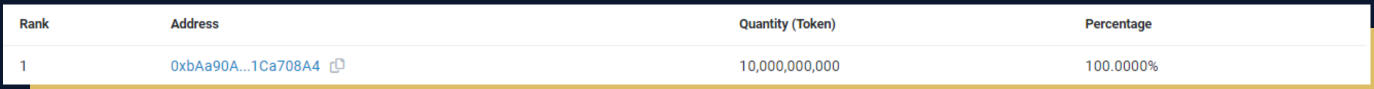
The top 10 holders collectively own 100.00% (10,000,000,000.00 Tokens) of Smurfs Coin

Token Total Supply: 10,000,000,000.00 Token | Total Token Holders: 1

Smurfs Coin Top 10 Token Holders

Source: BscScan.com

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



Rank	Address	Quantity (Token)	Percentage
1	0xbAa90A...1Ca708A4	10,000,000,000	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.