



SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



Scorpion
\$SCORP

08/04/2024

TOKEN OVERVIEW

Fees

- Buy fees: 0%
- Sell fees: 10%

Fees privileges

- Can change buy fees up to 10% and sell fees up to 10%

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
 - Can exclude / include from rewards
-

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

SCORP 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 **Scorpion** (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x66cEdcA3990b9a96369A58d08d62963c4e4eC779

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 **08/04/2024**



WEBSITE DIAGNOSTIC

<https://presale.scorpion.casino/>



0-49



50-89



90-100



Performance



Accessibility



Best
Practices



SEO



Progressive
Web App

Socials



Twitter

<https://twitter.com/ScorpionCasino>



Telegram

https://t.me/scorpioncasino_official

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	Low
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 addresses from transactions
- Contract owner can exclude/include wallet from rewards

```
function excludeFromReward(address account) public onlyOwner {
    require(!_isExcluded[account], "Account is already excluded");
    if (_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeInReward(address account) external onlyOwner {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- Contract owner can exclude/include wallet from tax

```
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
    _isExcludedFromBurnFee[account] = true;
}

function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
    _isExcludedFromBurnFee[account] = false;
}
```

- Contract owner can change marketingWallet address

Default value:

marketingWallet: 0xB87bc568B2182eEca900534Ed5e16506BF5084C6

```
function setMarketingWallet(address _addr) external onlyOwner {
    marketingWallet = _addr;
}
```

● Contract owner can change swap settings (without threshold)

```
function setNumTokensSellToAddToLiquidity(uint256 amount)
    external
    onlyOwner
{
    numTokensSellToAddToLiquidity = amount * 10 ** _decimals;
}

function setSwapAndLiquifyEnabled(bool _enabled) external onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}
```

● Contract owner can change buy fees up to 10% and sell fees up to 10%

```
function setSellFees(
    uint256 tFee,
    uint256 lFee,
    uint256 mFee,
    uint256 bFee
) external onlyOwner {
    uint256 toFees = tFee+lFee+mFee+bFee;
    require(toFees <= 10, "BEP20: Maximum BUY & SELL fees allowed is 10%");
    _sellRefFee = tFee;
    refFee = _sellRefFee;
    _sellLiquidityFee = lFee;
    liquidityFee = _sellLiquidityFee;
    _sellMarketingFee = mFee;
    marketingFee = _sellMarketingFee;
    _sellBurnFee = bFee;
    burnFee = _sellBurnFee;
}

function setBuyFees(
    uint256 tFee,
    uint256 lFee,
    uint256 mFee,
    uint256 bFee
) external onlyOwner {
    uint256 toFees = tFee+lFee+mFee+bFee;
    require(toFees <= 10, "BEP20: Maximum BUY & SELL fees allowed is 10%");
    _buyRefFee = tFee;
    refFee = _buyRefFee;
    _buyLiquidityFee = lFee;
    liquidityFee = _buyLiquidityFee;
    _buyMarketingFee = mFee;
    marketingFee = _buyMarketingFee;
    _buyBurnFee = bFee;
    burnFee = _buyBurnFee;
}
```

● Contract owner can transfer ownership

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

● Contract owner can renounce ownership

```
function renounceOwnership() public virtual 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 no HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees: 0%

Sell fees: 10%

Max TX: N/A

Max Wallet: 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



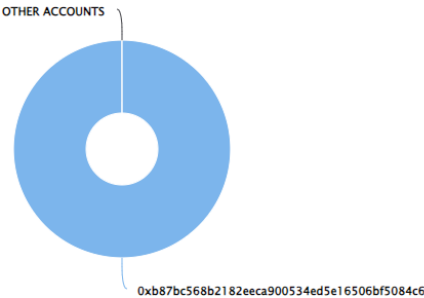
SCORP TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

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


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

Scorpion 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	0xB87bc568...6BF5084C6 	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.

