freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

ChampInu

$ChampInu

16/01/2022

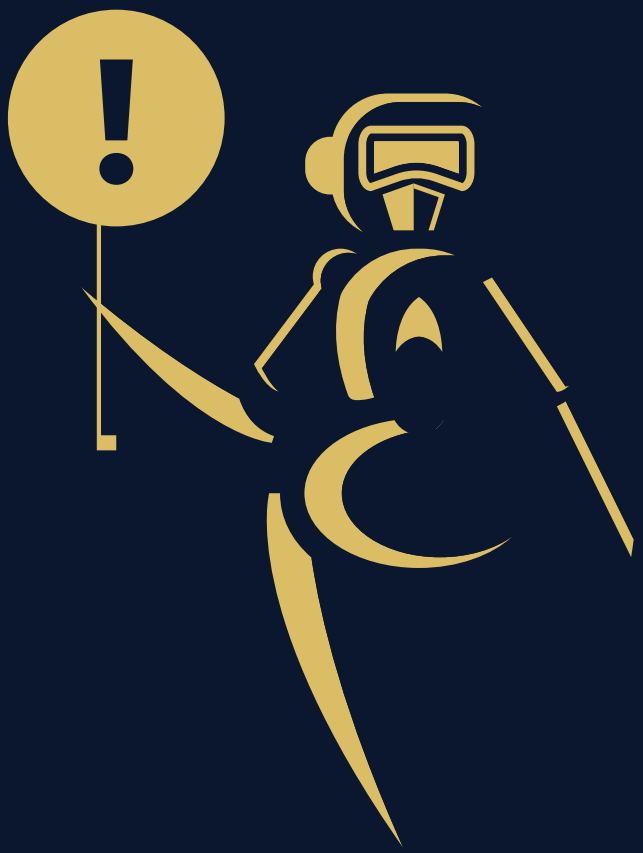
TABLE OF CONTENTS

DISCLAIMER

The information provided on this analysis decoument 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 ChampInu (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x44C263E76814dCf9c710E992306c4FdfBfB0429D

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 16/01/2022

WEBSITE DIAGNOSTIC

https://champinu.io


0-49


50-89


90-100

Performance

Accessability

Best Practices

SEO

Progressive Web App

Metrics


First Contentful Paint

2.5 s


Time to interactive

3.1 s


Speed Index

2.7 s


Total Blocking Time

270 ms


Large Contentful Paint

2.9 s


Cumulative Layout Shift

0

Website Improvements

Image elements do not have explicit width and height

Avoid enormous network payloads Total size was 4,739 KiB

Background and foreground colors do not have a sufficient contrast ratio

<html> element does not have a [lang] attribute

Links do not have a discernible name

AUDIT OVERVIEW

94

Security Score

92

Static Scan
Automatic scanning for common vulnerabilities

96

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	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 exclude/include wallet from fees

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

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

Contract owner can exclude/include wallet from reward

```
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 not excluded");
    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 swap

```
function excludeFromSwapAndLiquify(address account) public onlyOwner {
    _isExcludedFromSwapAndLiquify[account] = true;
}

function includeInSwapAndLiquify(address account) public onlyOwner {
    _isExcludedFromSwapAndLiquify[account] = false;
}
```

Contract owner can change max tx amount

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    _maxTxAmount = maxTxAmount;
}
```

Contract owner can renounce ownership

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

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 change the fees

```
function setReflectionFeePercent(uint256 reflectionFee) external onlyOwner() {
    _reflectionFee = reflectionFee;
}

function setMarketingFeePercent(uint256 marketingFee) external onlyOwner() {
    _marketingFee = marketingFee;
}

function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner {
    _liquidityFee = liquidityFee;
}
```

Contract owner can change sell fees

```
function setSellFee(uint256 sellReflectionFee, uint256 sellMarketingFee, uint256 sellLiquidityFee) external
onlyOwner {
    _sellReflectionFee = sellReflectionFee;
    _sellMarketingFee = sellMarketingFee;
    _sellLiquidityFee = sellLiquidityFee;
}
```

Contract owner can change BuyBack/AutoBuyBack and limits

```
function setBuyBackSellLimit(uint256 buyBackSellSetLimit) external onlyOwner {
    buyBackSellLimit = buyBackSellSetLimit;
}
.
.
.
function setBuyBackEnabled(bool _enabled) public onlyOwner {
    buyBackEnabled = _enabled;
    emit BuyBackEnabledUpdated(_enabled);
}

function setAutoBuyBackEnabled(bool _enabled) public onlyOwner {
    _isAutoBuyBack = _enabled;
    emit AutoBuyBackEnabledUpdated(_enabled);
}
```

Contract owner can change swap settings

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

Contract owner can change address for fees and sell fees

```
function setAddressFee(address _address, bool _enable, uint256 _addressReflectionFee, uint256 _addressLi-
quidityFee) external onlyOwner {
        _addressFees[_address].enable = _enable;
        _addressFees[_address]._reflectionFee = _addressReflectionFee;
        _addressFees[_address]._liquidityFee = _addressLiquidityFee;
}

function setSellAddressFee(address _address, bool _enable, uint256 _addressReflectionFee, uint256
_addressLiquidityFee) external onlyOwner {
        _addressFees[_address].enable = _enable;
        _addressFees[_address]._sellReflectionFee = _addressReflectionFee;
        _addressFees[_address]._sellLiquidityFee = _addressLiquidityFee;
}
```

Recomandation:

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 reviewd and analyzed with static tools.

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

Found no issue during the first review.

TOKEN DETAILS

Details

Buy fees:	12%
Sell fees:	13%
Max TX:	200000000000000000000000
Max Sell:	N/A

Honeypot Risk

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

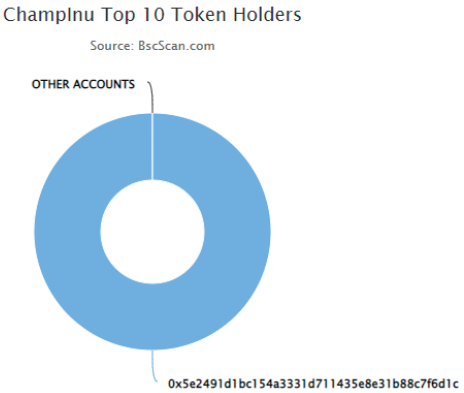
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

CHAMPINU TOKEN DISTRIBUTION & TOP 10 TOKEN HOLDERS

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

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



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

Rank	Address	Quantity (Token)	Percentage
1	0x5e2491d1bc154a3331d711435e8e31b88c7f6d1c	1,000,000,000,000,000	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.