freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

HIGHLANDER TOKEN

Highlander

$HLR

05/08/2023

TOKEN OVERVIEW

Fees

• Buy fees:	10%
• Sell fees:	25%

Fees privileges

• Can change buy fees up to 25%, sell fees up to 25% and transfer fees up to 25%

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

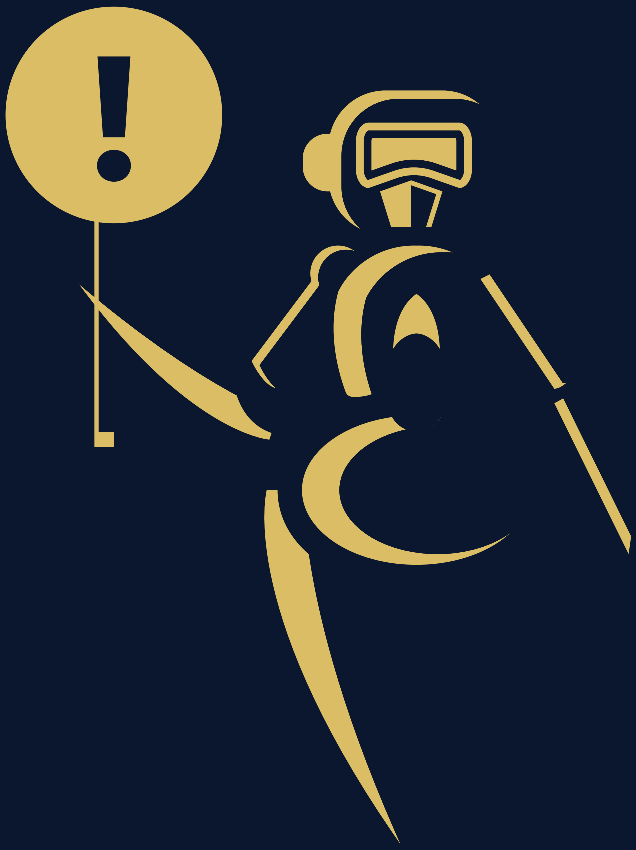
TABLE OF CONTENTS

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

0xe77CA10e5eC709eAE75925d9332C4bc792D25032

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 05/08/2023

WEBSITE DIAGNOSTIC

http://highlandertoken.com/

🟥 0-49 | 🟨 50-89 | 🟩 90-100

NA	NA	NA	NA	NA
Performance	Accessibility	Best Practices	SEO	Progressive Web App

Socials

Twitter	Telegram
N/A	N/A

AUDIT OVERVIEW

91

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

90

ERC Scan
Automatic checks for ERC's conformance

0	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	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 excludeFromFees(address account, bool isExcluded) public onlyOwner {
        isExcludedFromFees[account] = isExcluded;

        emit ExcludeFromFees(account, isExcluded);
}
```

● Contract owner can change marketingAddress and lpTokensReceiver addresses

Current values:

marketingAddress: 0xe54e971f35cbfecea76d3ef41b96ca9992b547b4

lpTokensReceiver: 0x0000000000000000000000000000000000000000

```
function marketingAddressSetup(address _newAddress) public onlyOwner {
        marketingAddress = _newAddress;

        excludeFromFees(_newAddress, true);

        emit marketingAddressUpdated(_newAddress);
}

function lpTokensReceiverSetup(address _newAddress) public onlyOwner {
        lpTokensReceiver = _newAddress;

        emit LpTokensReceiverUpdated(_newAddress);
}
```

● Contract owner can burn tokens

```
function burn(uint256 amount) public virtual {
        _burn(_msgSender(), amount);
}

function burnFrom(address account, uint256 amount) public virtual {
        _spendAllowance(account, _msgSender(), amount);
        _burn(account, amount);
}
```

● Contract owner can change buy fees up to 25%, sell fees up to 25% and transfer fees up to 25%

```
function liquidityFeesSetup(uint16 _buyFee, uint16 _sellFee, uint16 _transferFee) public onlyOwner {
      liquidityFees = [_buyFee, _sellFee, _transferFee];

      totalFees[0] = 0 + vaultFees[0] + marketingFees[0] + autoBurnFees[0] + liquidityFees[0];
      totalFees[1] = 0 + vaultFees[1] + marketingFees[1] + autoBurnFees[1] + liquidityFees[1];
      totalFees[2] = 0 + vaultFees[2] + marketingFees[2] + autoBurnFees[2] + liquidityFees[2];
      require(totalFees[0] <= 2500 && totalFees[1] <= 2500 && totalFees[2] <= 2500, "TaxesDefaultRouter:
Cannot exceed max total fee of 25%");

      emit liquidityFeesUpdated(_buyFee, _sellFee, _transferFee);
}

function marketingFeesSetup(uint16 _buyFee, uint16 _sellFee, uint16 _transferFee) public onlyOwner {
      marketingFees = [_buyFee, _sellFee, _transferFee];

      totalFees[0] = 0 + vaultFees[0] + marketingFees[0] + autoBurnFees[0] + liquidityFees[0];
      totalFees[1] = 0 + vaultFees[1] + marketingFees[1] + autoBurnFees[1] + liquidityFees[1];
      totalFees[2] = 0 + vaultFees[2] + marketingFees[2] + autoBurnFees[2] + liquidityFees[2];
      require(totalFees[0] <= 2500 && totalFees[1] <= 2500 && totalFees[2] <= 2500, "TaxesDefaultRouter:
Cannot exceed max total fee of 25%");

      emit marketingFeesUpdated(_buyFee, _sellFee, _transferFee);
}

function autoBurnFeesSetup(uint16 _buyFee, uint16 _sellFee, uint16 _transferFee) public onlyOwner {
      autoBurnFees = [_buyFee, _sellFee, _transferFee];

       totalFees[0] = 0 + vaultFees[0] + marketingFees[0] + autoBurnFees[0] + liquidityFees[0];
      totalFees[1] = 0 + vaultFees[1] + marketingFees[1] + autoBurnFees[1] + liquidityFees[1];
      totalFees[2] = 0 + vaultFees[2] + marketingFees[2] + autoBurnFees[2] + liquidityFees[2];
      require(totalFees[0] <= 2500 && totalFees[1] <= 2500 && totalFees[2] <= 2500, "TaxesDefaultRouter:
Cannot exceed max total fee of 25%");

      emit autoBurnFeesUpdated(_buyFee, _sellFee, _transferFee);
}

function vaultFeesSetup(uint16 _buyFee, uint16 _sellFee, uint16 _transferFee) public onlyOwner {
      vaultFees = [_buyFee, _sellFee, _transferFee];

       totalFees[0] = 0 + vaultFees[0] + marketingFees[0] + autoBurnFees[0] + liquidityFees[0];
      totalFees[1] = 0 + vaultFees[1] + marketingFees[1] + autoBurnFees[1] + liquidityFees[1];
      totalFees[2] = 0 + vaultFees[2] + marketingFees[2] + autoBurnFees[2] + liquidityFees[2];
      require(totalFees[0] <= 2500 && totalFees[1] <= 2500 && totalFees[2] <= 2500, "TaxesDefaultRouter:
Cannot exceed max total fee of 25%");

      emit vaultFeesUpdated(_buyFee, _sellFee, _transferFee);
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    _transferOwnership(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");
    _transferOwnership(newOwner);
}

function _transferOwnership(address newOwner) internal virtual {
    address oldOwner = _owner;
    _owner = newOwner;
    emit OwnershipTransferred(oldOwner, newOwner);
}
```

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:	25%
Transfer fees:	0%
Max TX:	N/A
Max Sell:	N/A

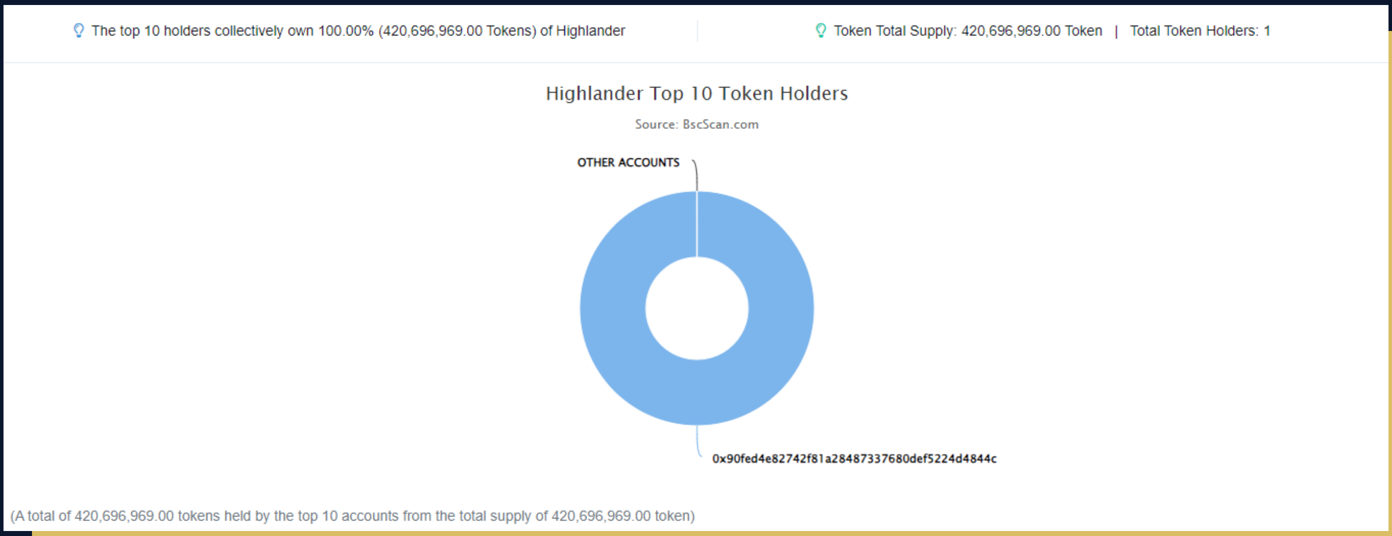
Honeypot Risk

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

HLR TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS



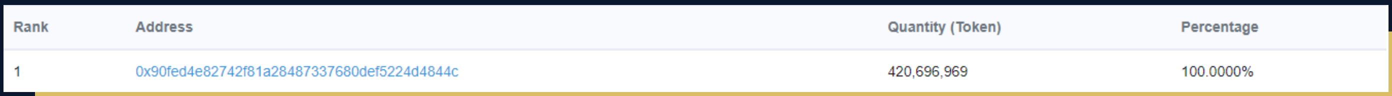
The top 10 holders collectively own 100.00% (420,696,969.00 Tokens) of Highlander

Token Total Supply: 420,696,969.00 Token | Total Token Holders: 1

Highlander Top 10 Token Holders

Source: BscScan.com

(A total of 420,696,969.00 tokens held by the top 10 accounts from the total supply of 420,696,969.00 token)



Rank	Address	Quantity (Token)	Percentage
1	0x90fed4e82742f81a28487337680def5224d4844c	420,696,969	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.