freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

ZAHNYMOUS

$ZAH

12/10/2022

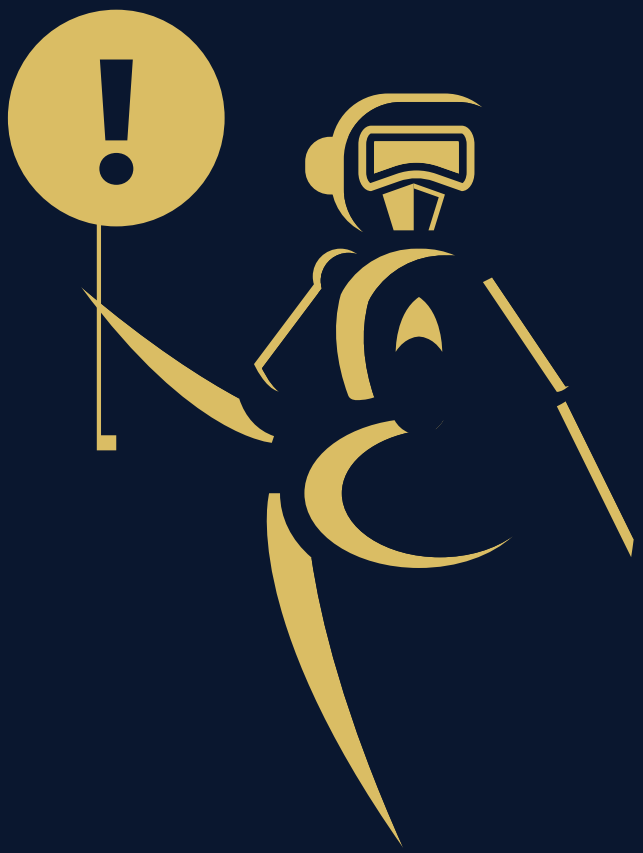
TABLE OF CONTENTS

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

0xE061156135B7b7847FD4dB74992ac8555C0CD5A7

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 12/10/2022

AUDIT OVERVIEW

99

Security Score
AUDIT RESULT: PASS

100

Static Scan
Automatic scanning for common vulnerabilities

98

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't exclude an address from transactions

Contract owner can exclude/include wallet from tax

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
    _isExcludedFromFees[account] = excluded;
    emit ExcludeFromFees(account, excluded);
}
```

Contract owner must enable trading (once enabled, can never be turned off)

```
function enableTrading() external onlyOwner {
    tradingActive = true;
    swapEnabled = true;
}
```

Contract owner can change swap settings

```
function updateSwapTokensAtAmount(uint256 newAmount) external onlyOwner returns (bool){
    require(newAmount >= totalSupply() * 1 / 100000, "Swap amount cannot be lower than 0.001% total
supply.");
    emit updatedTokensAtSwapAmount(newAmount, swapTokensAtAmount);
    swapTokensAtAmount = newAmount;
    return true;
}
```

Contract owner can change marketingWallet and devWallet addresses

Current values

marketingWallet : 0xd7da4746deb02d94c77ec19589929c5e4b323ee8

devWallet: 0x68c8840c1f8916bd0059be82b60d1ef858a26678

```
function updateMarketingWallet(address newMarketingWallet) external onlyOwner {
    require(newMarketingWallet != address(0), "Marketing wallet can not be set to a zero address");
    emit marketingWalletUpdated(newMarketingWallet, marketingWallet);
    marketingWallet = newMarketingWallet;
}

function updateDevWallet(address newWallet) external onlyOwner {
    emit devWalletUpdated(newWallet, devWallet);
    devWallet = newWallet;
}
```

Contract owner can change buy fees up to 20%

```
function updateBuyFees(uint256 _marketingFee, uint256 _liquidityFee, uint256 _devFee) external onlyOwner
{
        emit updatedBuyMarketingFee(_marketingFee, buyMarketingFee);
        emit updatedBuyLiquidityFee(_liquidityFee, buyLiquidityFee);
        emit updatedBuyDevFee(_devFee, buyDevFee);
        buyMarketingFee = _marketingFee;
        buyLiquidityFee = _liquidityFee;
        buyDevFee = _devFee;
        buyTotalFees = buyMarketingFee + buyLiquidityFee + buyDevFee;
        require(buyTotalFees <= 20, "Must keep fees at 20% or less");
}
```

Contract owner can change sell fees up to 20%

```
function updateSellFees(uint256 _marketingFee, uint256 _liquidityFee, uint256 _devFee) external onlyOwner
{
        emit updateSellMarketingFee(_marketingFee, sellMarketingFee);
        emit updateSellLiquidityFee(_liquidityFee, sellLiquidityFee);
        emit updateSellDevFee(_devFee, sellDevFee);
        sellMarketingFee = _marketingFee;
        sellLiquidityFee = _liquidityFee;
        sellDevFee = _devFee;
        sellTotalFees = sellMarketingFee + sellLiquidityFee + sellDevFee;
        require(sellTotalFees <= 20, "Must keep fees at 20% or less");
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public override onlyOwner {
        _isExcludedFromFees[_owner] = false;
        emit OwnershipTransferred(_owner, address(0));
        ZAHAntiBot(zahAntiBot).setTokenOwner(address(0));
        _owner = address(0);
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public override onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _isExcludedFromFees[_owner] = false;
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
        ZAHAntiBot(zahAntiBot).setTokenOwner(newOwner);
        _isExcludedFromFees[_owner] = true;
}
```

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 issue during the first review.

TOKEN DETAILS

Details

Buy fees:	5%
Sell fees:	13%
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

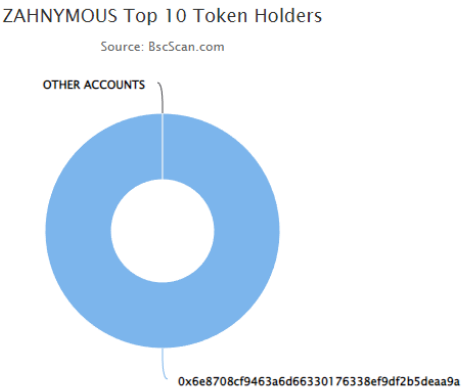
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

ZAHNYMOUS TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (100,000,000.00 Tokens) of ZAHNYMOUS

Token Total Supply: 100,000,000.00 Token | Total Token Holders: 1



(A total of 100,000,000.00 tokens held by the top 10 accounts from the total supply of 100,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x6e8708cf9463a6d66330176338ef9df2b5deaa9a	100,000,000	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.