freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

GuardAI

$GuardAI

22/03/2024

TOKEN OVERVIEW

Fees

- Buy fees: 20%
- Sell fees: 20%

Fees privileges

- Can change buy fees up to 20% and sell fees up to 20%

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

- Can change max tx amount and max wallet amount (with threshold)

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude / include from fees

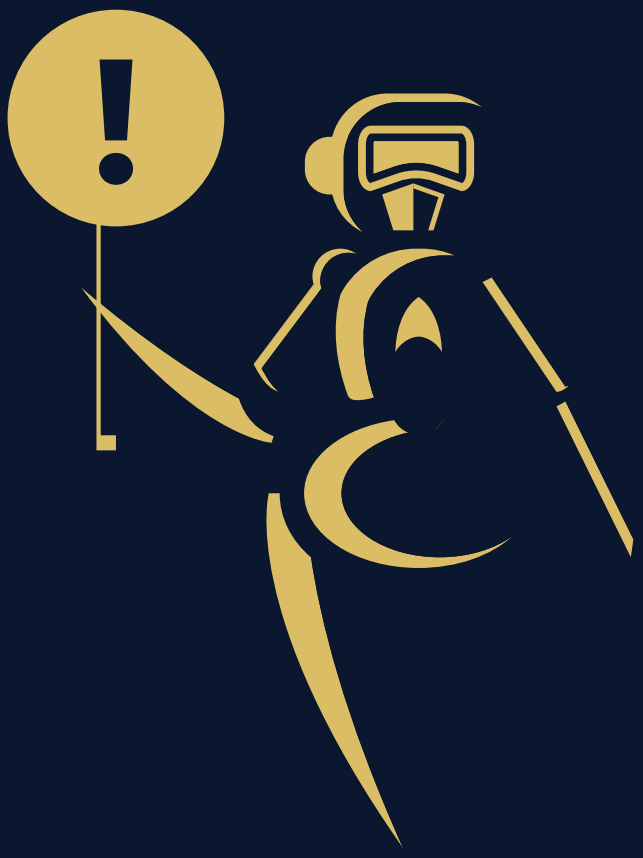
TABLE OF CONTENTS

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

0xd1679946ba555eBF5cb38e8b089EF1E1e5D2ABB1

Network: Ethereum (ETH)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 22/03/2024

WEBSITE DIAGNOSTIC

https://www.buyguard.io/

0-49 50-89 90-100

92	95	91	92	NA
Performance	Accessibility	Best Practices	SEO	Progressive Web App

Socials

Twitter

https://twitter.com/BuyGuardBot

Telegram

https://t.me/GuardAICommunity

AUDIT OVERVIEW

86

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

84

ERC Scan
Automatic checks for ERC's conformance

0	High
2	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	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 tax

```
function excludeFromFee(address account, bool isEx) external onlyOwner {
        require(isExcludedFromFee[account] != isEx, "already");
        isExcludedFromFee[account] = isEx;
        emit ExcludedFromFee(account, isEx);
}
```

- Contract owner can exclude/include wallet from tx limitations

```
function excludeFromMaxTransactionAmount(address account, bool isEx)
        external
        onlyOwner
    {
        require(isExcludedFromMaxTransactionAmount[account]!=isEx, "already");
        isExcludedFromMaxTransactionAmount[account] = isEx;
        emit ExcludedFromMaxTransactionAmount(account, isEx);
}
```

- Contract owner can change max wallet amount limitation (with threshold)

```
function updateMaxWallet(uint256 _maxWallet) external onlyOwner {
        require(_maxWallet>=totalSupply() / 10000, "maxWallet >= total supply / 10000");
        emit UpdateMaxWallet(_maxWallet, maxWallet);
        maxWallet = _maxWallet;
}
```

- Contract owner can change max tx amount limitation (with threshold)

```
function updateMaxTransactionAmount(uint256 _maxTransactionAmount)
        external
        onlyOwner
    {
        require(_maxTransactionAmount>=totalSupply() / 10000, "maxTransactionAmount >= total supply /
10000");
        emit UpdateMaxTransactionAmount(_maxTransactionAmount, maxTransactionAmount);
        maxTransactionAmount = _maxTransactionAmount;
}
```

Contract owner can change marketingWallet address

Current value:

marketingWallet: 0x4A627778AdF4eB8B07B651E20F94Cc92adDD4210

```
function updateMarketingWallet(
      address _marketingWallet,
      bool _isMarketingFeeBaseToken
   ) external onlyOwner {
      require(_marketingWallet != address(0), "marketing wallet can't be 0");
      emit UpdateMarketingWallet(_marketingWallet, _isMarketingFeeBaseToken,
         marketingWallet, isMarketingFeeBaseToken);
      marketingWallet = _marketingWallet;
      isMarketingFeeBaseToken = _isMarketingFeeBaseToken;
      isExcludedFromFee[_marketingWallet] = true;
      isExcludedFromMaxTransactionAmount[_marketingWallet] = true;
}
```

Contract owner can change swap settings (with threshold)

```
function updateMinAmountToTakeFee(uint256 _minAmountToTakeFee)
      external
      onlyOwner
   {
      require(_minAmountToTakeFee > 0, "minAmountToTakeFee > 0");
      emit UpdateMinAmountToTakeFee(_minAmountToTakeFee, minAmountToTakeFee);
      minAmountToTakeFee = _minAmountToTakeFee;
}
```

Contract owner can change buy fees up to 20% and sell fees up to 20%

```
function updateLiquidityFee(uint16 _sellLiquidityFee, uint16 _buyLiquidityFee) external onlyOwner {
      require(_sellLiquidityFee + (sellMarketingFee) <= 200, "sell fee <= 20%");
      require(_buyLiquidityFee + (buyMarketingFee) <= 200, "buy fee <= 20%");
      emit UpdateLiquidityFee( _sellLiquidityFee, _buyLiquidityFee, sellLiquidityFee, buyLiquidityFee );
      sellLiquidityFee = _sellLiquidityFee;
      buyLiquidityFee = _buyLiquidityFee;
}

function updateMarketingFee(uint16 _sellMarketingFee, uint16 _buyMarketingFee) external onlyOwner {
      require( _sellMarketingFee + (sellLiquidityFee) <= 200, "sell fee <= 20%" );
      require(_buyMarketingFee + (buyLiquidityFee) <= 200, "buy fee <= 20%");
      emit UpdateMarketingFee( _sellMarketingFee, _buyMarketingFee, sellMarketingFee, buyMarketingFee);
      sellMarketingFee = _sellMarketingFee;
      buyMarketingFee = _buyMarketingFee;
}
```

Contract owner has ability to retrieve any token held by the contract

Native tokens excluded

```
function withdrawETH() external onlyOwner {
    (bool success, )=address(owner()).call{value: address(this).balance}("");
    require(success, "Failed in withdrawal");
}

function withdrawToken(address token) external onlyOwner{
    require(address(this) != token, "Not allowed");
    IERC20(token).safeTransfer(owner(), IERC20(token).balanceOf(address(this)));
}
```

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);
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    _transferOwnership(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:	20%
Sell fees:	20%
Max TX:	20,000
Max Wallet:	20,000

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:	100% unlocked tokens

GUARDAI TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

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

Token Total Supply: 1,000,000.00 Token | Total Token Holders: 2

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

Rank	Address	Quantity (Token)	Percentage
1	0x47d06007...a454e0aA8	999,999.999	100.0000%
2	0xBAdCbdDe...e3CAf0180	0.001	0.0000%

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.