freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

PEPESHIB

$PEPESHIB

21/03/2024

TOKEN OVERVIEW

Fees

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

Fees privileges

- Can change fees up to 100%

Ownership

- Owned

Minting

- Mint function detected

Max Tx Amount / Max Wallet Amount

- Can’t change max tx amount and max wallet amount

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude / include from fees

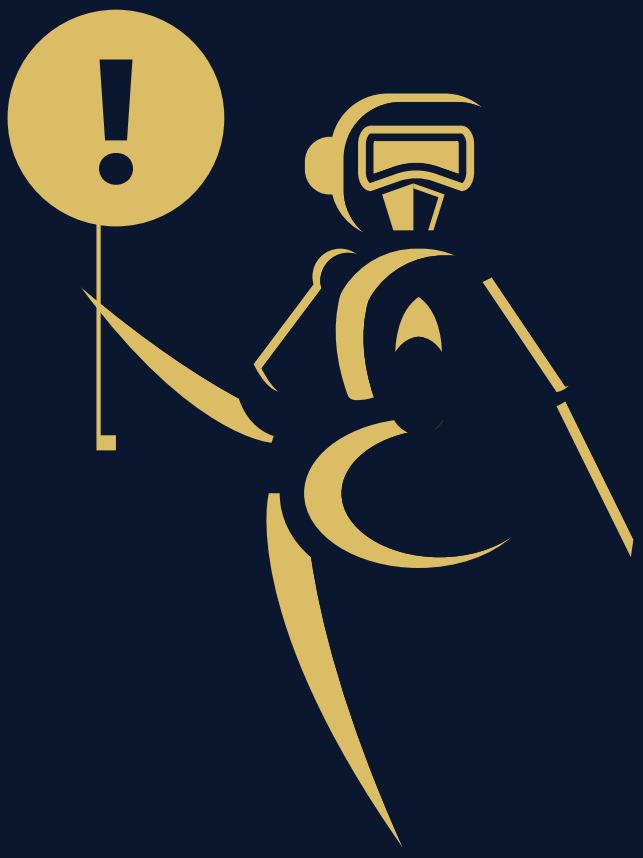
TABLE OF CONTENTS

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

0x6E1b174EcCd7e857E9cA280B46081c5e9556f94A

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 21/03/2024

WEBSITE DIAGNOSTIC

https://pepeshib.lol/


0-49


50-89


90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials


Twitter

https://twitter.com/PepeShib_erc20


Telegram

https://t.me/PepeShibPortal

AUDIT OVERVIEW

32

Security Score
HIGH RISK
Audit FAIL

93

Static Scan
Automatic scanning for common vulnerabilities

31

ERC Scan
Automatic checks for ERC's conformance

2	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	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

Can mint tokens after initial contract deploy

These functions together provide the functionality to mint tokens, with the option to use reflection mechanics if the token is configured as a reflection token.
There are a few potential risks associated with the provided mint function:

Unchecked Overflow: The line require(totalSupply() + amount <= maxSupply, "TokenFiERC20: max supply exceeded"); aims to prevent the total supply from exceeding the maxSupply. However, this line does not account for potential integer overflow. If totalSupply() is close to the maximum value of a uint256, adding amount could result in an overflow, causing unexpected behavior and potentially allowing the minting of more tokens than intended

Access Control Vulnerability: The onlyRole(DEFAULT_ADMIN_ROLE) modifier restricts access to this function to only those with the DEFAULT_ADMIN_ROLE role. However, if this role is not properly managed or if there is a vulnerability in the role management system, unauthorized users may gain access to the mint function, allowing them to create new tokens or exceed the maxSupply

```
function _mintReflection(address account, uint256 amount) private {
    // increase total amounts
    uint256 _rAmount = amount * _getRate();
    totalReflection.t = totalReflection.t + amount;
    totalReflection.r = totalReflection.r + _rAmount;

    // increase tBalance if the receiver address is excluded
    if (isExcludedFromReflectionRewards(account)) {
      _tOwned[account] = _tOwned[account] + amount;
    }
    _rOwned[account] = _rOwned[account] + _rAmount;

    emit Transfer(address(0), account, amount);
}

function mint(address to, uint256 amount) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(totalSupply() + amount <= maxSupply, "TokenFiERC20: max supply exceeded");
    if (isReflectionToken) {
      _mintReflection(to, amount);
    } else {
      super._mint(to, amount);
    }
}
```

Contract owner can change fees up to 100%

uint256 public constant MULTIPLIER_BASIS = 1e4;

```
function updateFees(ITokenLauncherERC20.Fees memory _fees) external onlyRole(DEFAULT_ADMIN_ROLE) {
    if (isReflectionToken) {
        require(_fees.reflection.percentage > 0, "TokenFiERC20: reflection percentage must be non-zero");
    } else {
        require(_fees.reflection.percentage == 0, "TokenFiERC20: reflection percentage must be zero");
    }
    uint256 maxFee = _fees.transferFee.percentage + _fees.burn.percentage + _fees.reflection.percentage +
_fees.buyback.percentage;
    require(maxFee <= MULTIPLIER_BASIS, "TokenFiERC20: fees sum must be less than 100%");
    fees = _fees;
}
```

Contract owner can exclude/include wallet from tax

```
function addExemptAddress(address account) external onlyRole(FEE_MANAGER_ROLE) {
    _exemptedFromTax.add(account);
    emit ExemptedAdded(account);
}

function removeExemptAddress(address account) external onlyRole(FEE_MANAGER_ROLE) {
    _exemptedFromTax.remove(account);
    emit ExemptedRemoved(account);
}
```

Contract owner can exclude/include wallet from reflection

Please note that isReflectionToken value is False

```
function excludeAccount(address account) external onlyReflection onlyFromAdminOrLauncher {
    require(!isExcludedFromReflectionRewards(account), "Account is already excluded");
    if (_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcludedFromReflectionRewards[account] = true;
    _excluded.push(account);
}

function includeAccount(address account) external onlyReflection onlyFromAdminOrLauncher {
    require(isExcludedFromReflectionRewards(account), "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            uint256 currentRate = _getRate();
            totalReflection.r = totalReflection.r - _rOwned[account];
            _rOwned[account] = _tOwned[account] * currentRate;
            _tOwned[account] = 0;
            totalReflection.r = totalReflection.r + _rOwned[account];

            _isExcludedFromReflectionRewards[account] = false;
            _excluded[i] = _excluded[_excluded.length - 1];
            _excluded.pop();
            break;
        }
    }
}
```

Contract owner can change buy back settings

```
uint256 public constant MULTIPLIER_BASIS = 1e4;

function setBuybackDetails(ITokenLauncherLiquidityPoolFactory.BuyBackDetails memory _buybackDetails)
external onlyRole(FEE_MANAGER_ROLE) {
    if (fees.buyback.percentage > 0) {
      require(_buybackDetails.liquidityBasisPoints <= MULTIPLIER_BASIS, "TokenFiERC20: liquidityBasisPoints
must be less than 10,000");
      require(_buybackDetails.priceImpactBasisPoints <= MULTIPLIER_BASIS, "TokenFiERC20: priceImpactBa-
sisPoints must be less than 10,000");
      require(_buybackDetails.router != address(0), "TokenFiERC20: router cannot be empty");
      require(_buybackDetails.pairToken != address(0), "TokenFiERC20: pairToken cannot be empty");
    }
    buybackDetails = _buybackDetails;

    emit BuyBackDetailsUpdated(
      _buybackDetails.router,
      _buybackDetails.pairToken,
      _buybackDetails.liquidityBasisPoints,
      _buybackDetails.priceImpactBasisPoints
    );
}
```

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 2 HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	0%
Sell fees:	0%
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

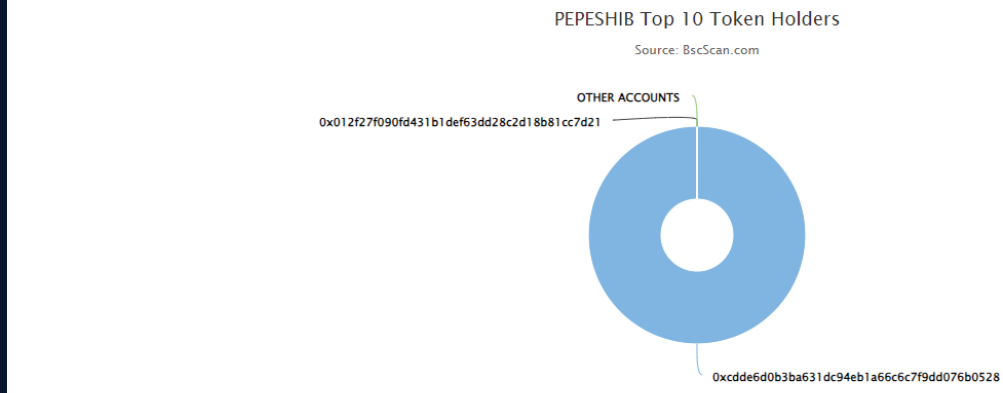
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

PEPESHIB TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

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

Token Total Supply: 100,000,000.00 Token | Total Token Holders: 2



(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	0xCDDe6d0B...D076b0528	99,999,999.406	100.0000%
2	0x012F27f0...b81cc7D21	0.594	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.