expelee

A Secure Place For Web3

SMART CONTRACT AUDIT OF

MyChat Presale

Contract Address

0x0664A1a6e140777CD529637a479e52C111c37826

Audit Summary

Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result: **PASSED**

Ownership: NOT **RENOUNCED**

KYC Verification: Not done till date of audit

Audit Date: 14/07/2022

Audit Team: **EXPELEE**

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.

DISCLAMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always Do your own research and protect yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.

Contract Review

Contract Name	**MyChat**
Compiler Version	**v0.8.7+commit.e28d00a7**
Optimization	**No with 200 runs**
License	**None license**
Explorer	**https://bscscan.com/address/0x0664A1a6e140777CD529637a479e52C111c37826#contracts**
Symbol	**MC**
Decimals	**18**
Total Supply	**100,000,000**
Domain	**https://www.mychat.life/**

Project Review

Token Name: MyChat

Web Site: https://www.mychat.life/

Twitter: https://twitter.com/mychattoken/

Telegram: https://t.me/MyChatENofficial

Contract Address:
0x0664A1a6e140777CD529637a479e52C111c37826

Platform: Binance Smart Chain

Token Type: BEP 20

Language: SOLIDITY

Audit Methodology

The scope of this report is to audit the smart contract source code. We have scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

Smart Contract Vulnerabilities	- Unhandled Exceptions - Transaction Order Dependency - Integer Overflow - Unrestricted Action - Incorrect Inheritance Order - Typographical Errors - Requirement Violation
Source Code Review	- Gas Limit and Loops - Deployment Consistency - Repository Consistency - Data Consistency - Token Supply Manipulation
Functional Assessment	- Operations Trail & Event Generation - Assets Manipulation - Liquidity Access

Vulnerability Checklist

№	Description.	Result
1	Compiler warnings.	Passed
2	Race conditions and Re-entrancy. Cross-function raceconditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module.	Passed
21	Fallback function security.	Passed

Manual Audit

🟢**Low-Risk**

🟢3 low-risk code issues found

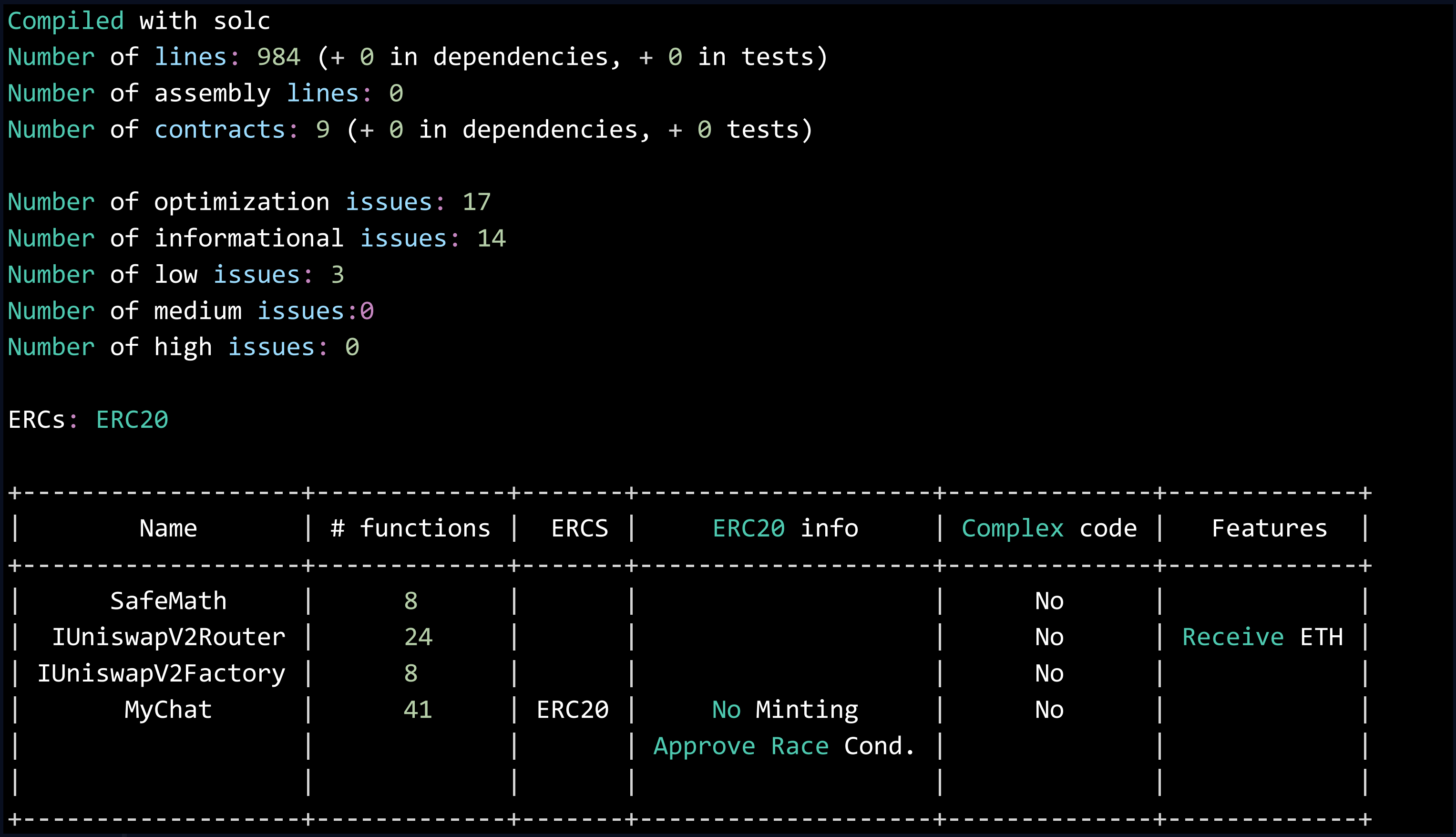
🟡**Medium-Risk**

🟡0 medium-risk code issues found

🔴**High-Risk**

🔴 0 high-risk code issues found

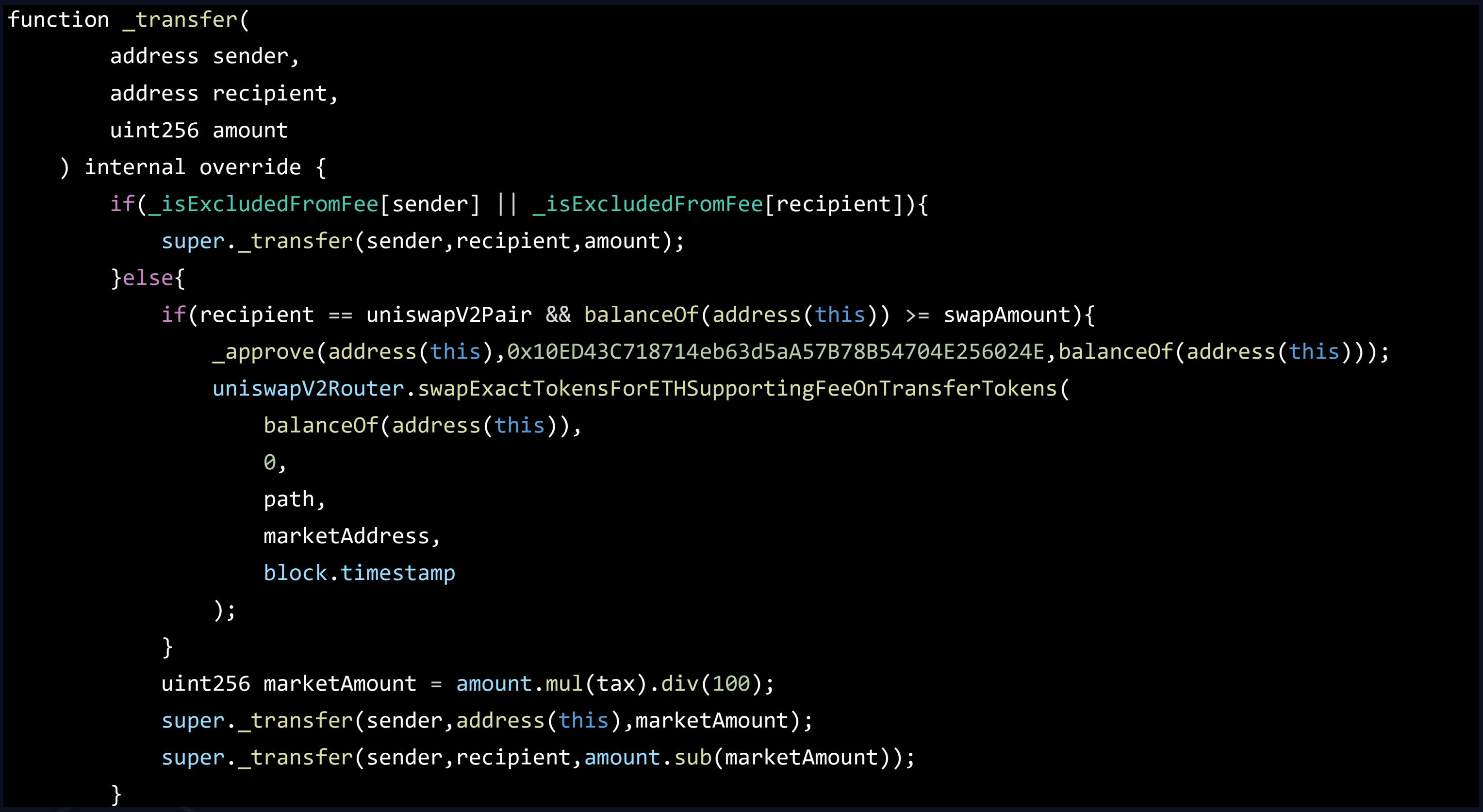
Audit Summary



```
Compiled with solc
Number of lines: 984 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 9 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 17
Number of informational issues: 14
Number of low issues: 3
Number of medium issues:0
Number of high issues: 0

ERCs: ERC20

+-------------------+-------------+-------+--------------------+--------------+-------------+
|        Name       | # functions |  ERCS |     ERC20 info     | Complex code |   Features  |
+-------------------+-------------+-------+--------------------+--------------+-------------+
|      SafeMath     |      8      |       |                    |      No      |             |
|  IUniswapV2Router |      24     |       |                    |      No      | Receive ETH |
| IUniswapV2Factory |      8      |       |                    |      No      |             |
|       MyChat      |      41     | ERC20 |     No Minting     |      No      |             |
|                   |             |       | Approve Race Cond. |              |             |
|                   |             |       |                    |              |             |
+-------------------+-------------+-------+--------------------+--------------+-------------+
```

Low-Risk

1) Reentrancy Vulnerabilities



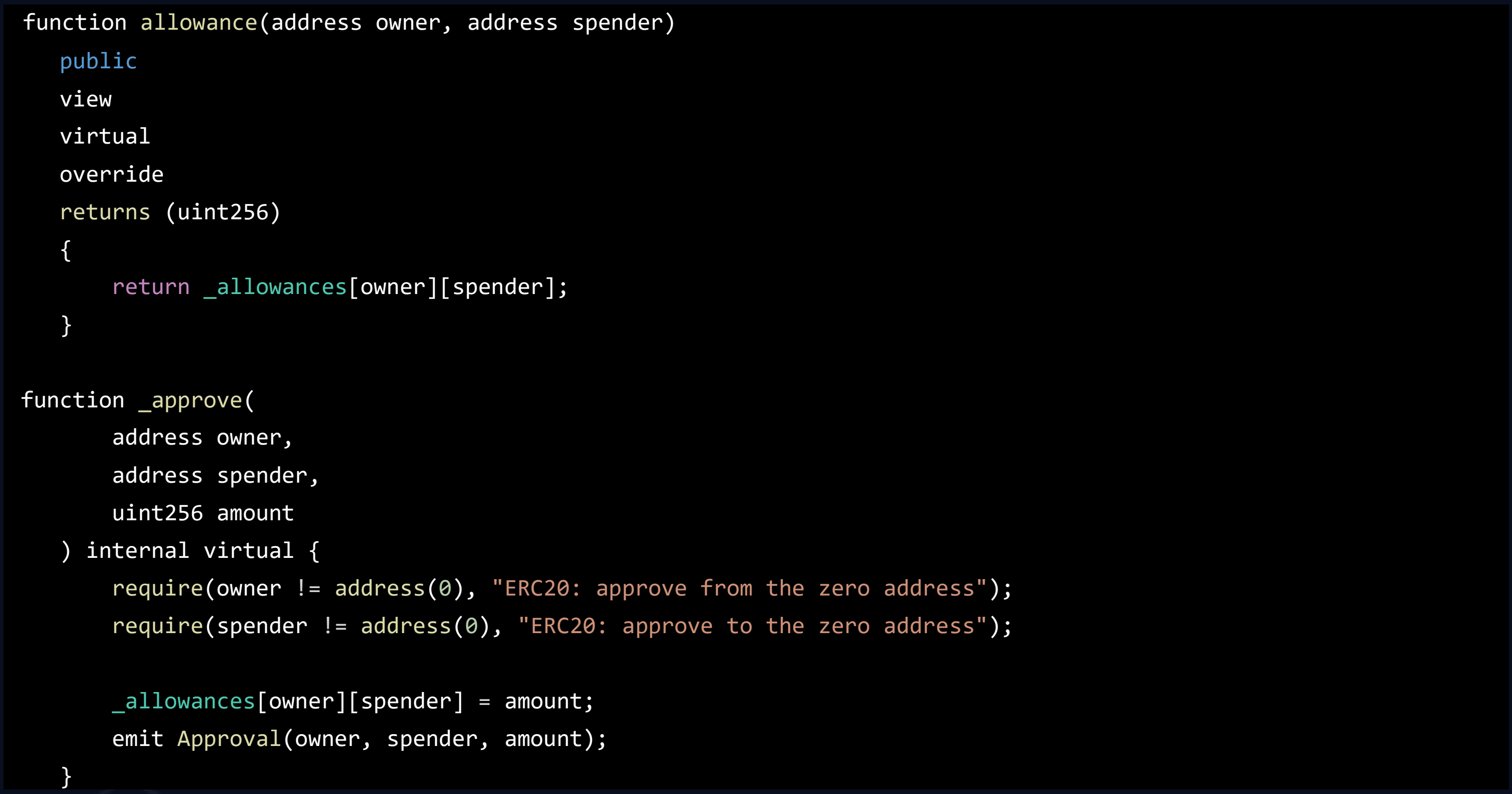
```
function _transfer(
        address sender,
        address recipient,
        uint256 amount
    ) internal override {
        if(_isExcludedFromFee[sender] || _isExcludedFromFee[recipient]){
            super._transfer(sender,recipient,amount);
        }else{
            if(recipient == uniswapV2Pair && balanceOf(address(this)) >= swapAmount){
                _approve(address(this),0x10ED43C718714eb63d5aA57B78B54704E256024E,balanceOf(address(this)));
                uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
                    balanceOf(address(this)),
                    0,
                    path,
                    marketAddress,
                    block.timestamp
                );
            }
            uint256 marketAmount = amount.mul(tax).div(100);
            super._transfer(sender,address(this),marketAmount);
            super._transfer(sender,recipient,amount.sub(marketAmount));
        }
```

Recommendation

Apply the check-effects-interactions function.

2) Local variable shadowing

Detection of shadowing using local variables .

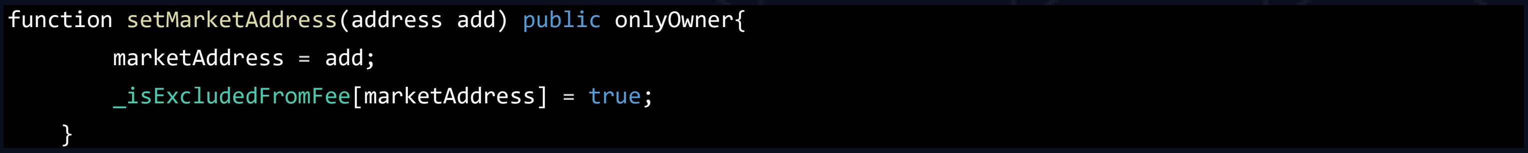


```
function allowance(address owner, address spender)
    public
    view
    virtual
    override
    returns (uint256)
    {
        return _allowances[owner][spender];
    }

function _approve(
        address owner,
        address spender,
        uint256 amount
    ) internal virtual {
        require(owner != address(0), "ERC20: approve from the zero address");
        require(spender != address(0), "ERC20: approve to the zero address");

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }
```

Recommendation

Rename the local variable that shadow another variable.

3) Missing zero address validation

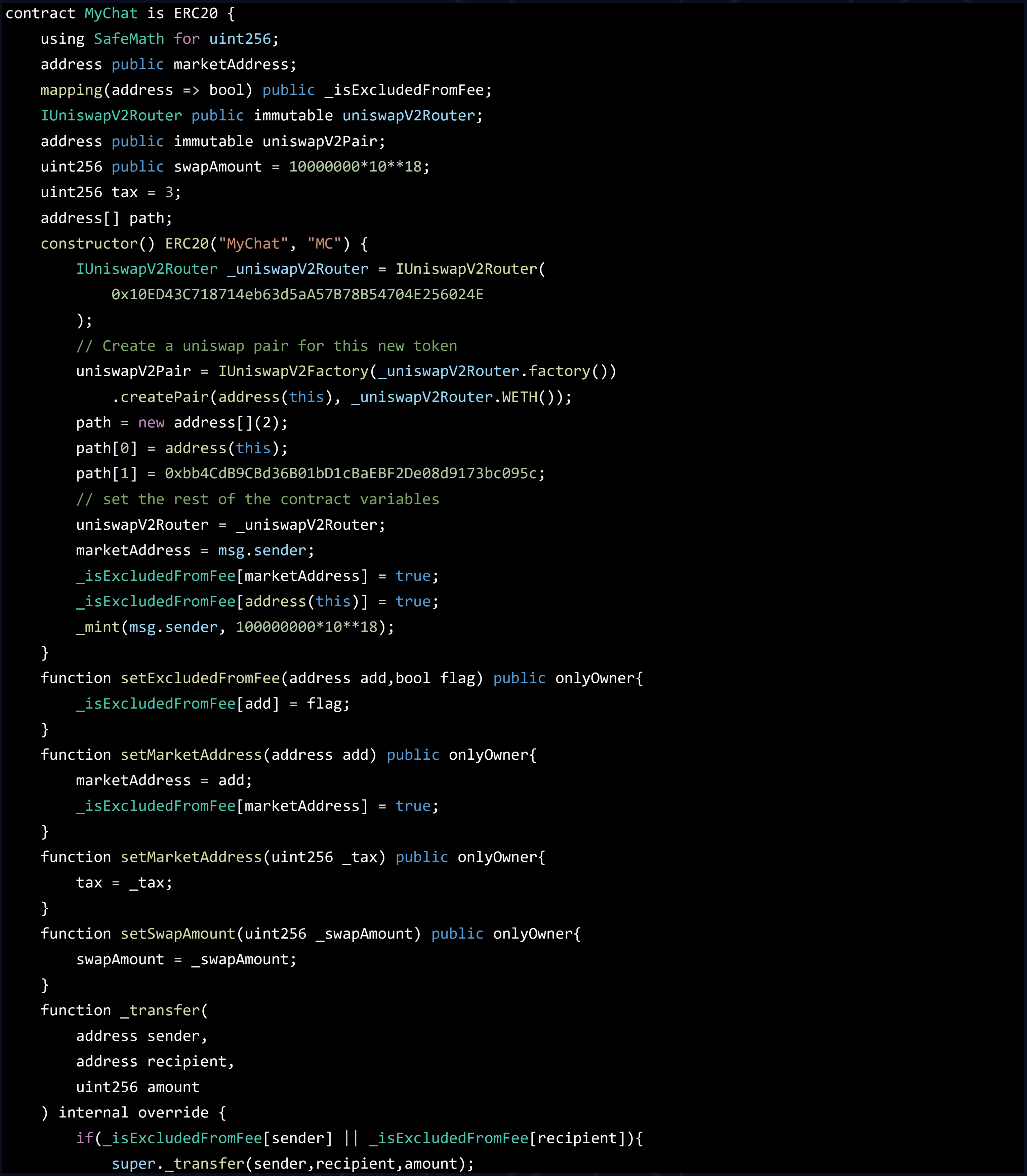


```
function setMarketAddress(address add) public onlyOwner{
        marketAddress = add;
        _isExcludedFromFee[marketAddress] = true;
    }
```

Recommendation

Check that the address is not zero.

Manual Audit (Contract Function)



```
contract MyChat is ERC20 {
    using SafeMath for uint256;
    address public marketAddress;
    mapping(address => bool) public _isExcludedFromFee;
    IUniswapV2Router public immutable uniswapV2Router;
    address public immutable uniswapV2Pair;
    uint256 public swapAmount = 10000000*10**18;
    uint256 tax = 3;
    address[] path;
    constructor() ERC20("MyChat", "MC") {
        IUniswapV2Router _uniswapV2Router = IUniswapV2Router(
            0x10ED43C718714eb63d5aA57B78B54704E256024E
        );
        // Create a uniswap pair for this new token
        uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());
        path = new address[](2);
        path[0] = address(this);
        path[1] = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
        // set the rest of the contract variables
        uniswapV2Router = _uniswapV2Router;
        marketAddress = msg.sender;
        _isExcludedFromFee[marketAddress] = true;
        _isExcludedFromFee[address(this)] = true;
        _mint(msg.sender, 100000000*10**18);
    }
    function setExcludedFromFee(address add,bool flag) public onlyOwner{
        _isExcludedFromFee[add] = flag;
    }
    function setMarketAddress(address add) public onlyOwner{
        marketAddress = add;
        _isExcludedFromFee[marketAddress] = true;
    }
    function setMarketAddress(uint256 _tax) public onlyOwner{
        tax = _tax;
    }
    function setSwapAmount(uint256 _swapAmount) public onlyOwner{
        swapAmount = _swapAmount;
    }
    function _transfer(
        address sender,
        address recipient,
        uint256 amount
    ) internal override {
        if(_isExcludedFromFee[sender] || _isExcludedFromFee[recipient]){
            super._transfer(sender,recipient,amount);
```

Important Points To Consider

✓ Verified contract source

✓ Token is sellable (not a honeypot) at this time

✗ Ownership renounced or source does not contain an owner contract

✗ Source does not contain a fee modifier

✓ Buy fee is less than 10% (3%)

✓ Sell fee is less than 10% (3%)

✗ Owner/creator wallet contains less than 10% of circulating token supply (56.05%)

About Expelee

Expelee is a community driven organisation dedicated to fostering an anti-rug movement. We're here to keep investment safe from fraudsters. We've encountered several rug pulls and know how it feels to be duped, which is why we don't want anybody else to go through the same experience. We are here to raise awareness through our services so that the future of cryptocurrency can be rug-free.

The auditing process focuses to the following considerations with collaboration of an expert team:

- Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
- Manually detailed examination of the code line by line by experts.
- Live test by multiple clients using Test net.
- Analysing failure preparations to check how the Smart
- Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

Social Media

/Expelee

/ExpeleeOfficial

/expelee-co