Audit Report for **Pepzilla**
Date: 16 March 2024
Audit result: **Passed with high risk.**

Token Address: 0x9515903090cE3aB282C86947eeA0cFE2cf3e5219

Name: Pepzilla

Symbol: PEPZ

Decimals: 9

Network: BscScan

Token Type: BEP-20

Owner: 0xeEF82F3a26AB31f3EC3047d5f455Cf76C7dA95c9

Deployer: 0xeEF82F3a26AB31f3EC3047d5f455Cf76C7dA95c9

Token Supply: 100000000000

Checksum: Bc6659e84744e0102ab19c1d1e78a22a

Testnet:
https://testnet.bscscan.com/address/0x65d1041e604a3fb92c5401744c3c46b01a22bd1a#code

Token Overview:

Buy Fee: 5-100%

Sell Fee: 5-100%

Transfer Fee: 0-0%

Fee Privilege: Owner

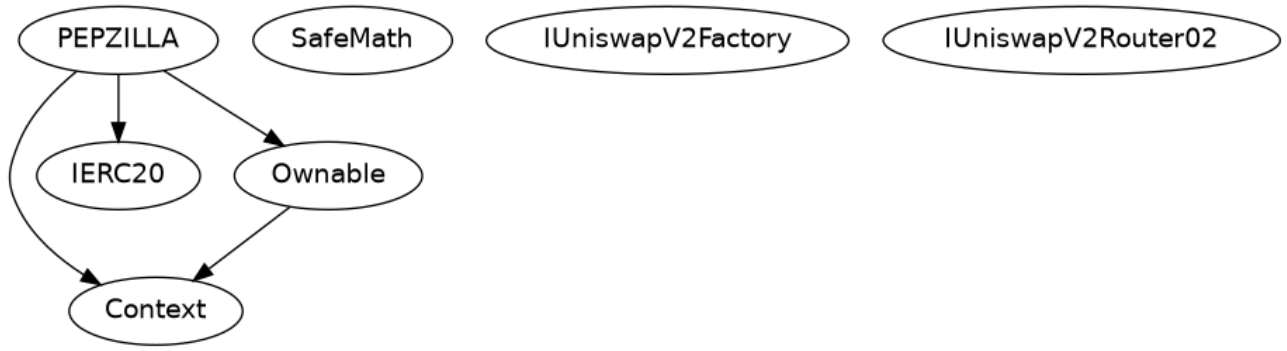
Ownership: Owned

Minting: None

Max Tx: Yes

Blacklist: Yes

Inheritance Tree



Static Analysis

A static analysis of the code was performed using Slither. No issues were found.

```
INFO:Detectors:
PEPZILLA.allowance(address,address).owner (PEPZILLA.sol#255) shadows:
        - Ownable.owner() (PEPZILLA.sol#56-58) (function)
PEPZILLA._approve(address,address,uint256).owner (PEPZILLA.sol#319) shadows:
        - Ownable.owner() (PEPZILLA.sol#56-58) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
PEPZILLA.setFee(uint256,uint256,uint256,uint256) (PEPZILLA.sol#553-558) should emit an event for:
        - _redisFeeOnBuy = redisFeeOnBuy (PEPZILLA.sol#554)
        - _redisFeeOnSell = redisFeeOnSell (PEPZILLA.sol#555)
        - _taxFeeOnBuy = taxFeeOnBuy (PEPZILLA.sol#556)
        - _taxFeeOnSell = taxFeeOnSell (PEPZILLA.sol#557)
PEPZILLA.setMinSwapTokensThreshold(uint256) (PEPZILLA.sol#561-563) should emit an event for:
        - _swapTokensAtAmount = swapTokensAtAmount (PEPZILLA.sol#562)
PEPZILLA.setMaxTxnAmount(uint256) (PEPZILLA.sol#571-573) should emit an event for:
        - _maxTxAmount = maxTxAmount (PEPZILLA.sol#572)
PEPZILLA.setMaxWalletSize(uint256) (PEPZILLA.sol#575-577) should emit an event for:
        - _maxWalletSize = maxWalletSize (PEPZILLA.sol#576)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Reentrancy in PEPZILLA._transfer(address,address,uint256) (PEPZILLA.sol#329-391):
        External calls:
        - swapTokensForEth(contractTokenBalance) (PEPZILLA.sol#361)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (PEPZILLA.sol#398-404
)
        External calls sending eth:
        - sendETHToFee(address(this).balance) (PEPZILLA.sol#364)
                - _marketingAddress.transfer(amount) (PEPZILLA.sol#408)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (PEPZILLA.sol#390)
                - _previousredisFee = _redisFee (PEPZILLA.sol#306)
        - _tokenTransfer(from,to,amount,takeFee) (PEPZILLA.sol#390)
                - _previoustaxFee = _taxFee (PEPZILLA.sol#307)
        - _redisFee = _redisFeeOnBuy (PEPZILLA.sol#378)
        - _redisFee = _redisFeeOnSell (PEPZILLA.sol#384)
        - _tokenTransfer(from,to,amount,takeFee) (PEPZILLA.sol#390)
                - _redisFee = _previousredisFee (PEPZILLA.sol#314)
                - _redisFee = 0 (PEPZILLA.sol#309)
```

```
INFO:Detectors:
PEPZILLA._redisFee (PEPZILLA.sol#181) is set pre-construction with a non-constant function or state variable:
        - _redisFeeOnSell
PEPZILLA._taxFee (PEPZILLA.sol#182) is set pre-construction with a non-constant function or state variable:
        - _taxFeeOnSell
PEPZILLA._previousredisFee (PEPZILLA.sol#184) is set pre-construction with a non-constant function or state variable:
        - _redisFee
PEPZILLA._previoustaxFee (PEPZILLA.sol#185) is set pre-construction with a non-constant function or state variable:
        - _taxFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
INFO:Detectors:
Pragma version^0.8.17 (PEPZILLA.sol#9) allows old versions
solc-0.8.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Router02.WETH() (PEPZILLA.sol#140) is not in mixedCase
Parameter PEPZILLA.setTrading(bool)._tradingOpen (PEPZILLA.sol#411) is not in mixedCase
Parameter PEPZILLA.toggleSwap(bool)._swapEnabled (PEPZILLA.sol#566) is not in mixedCase
Constant PEPZILLA._name (PEPZILLA.sol#163) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PEPZILLA._symbol (PEPZILLA.sol#164) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PEPZILLA._decimals (PEPZILLA.sol#165) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PEPZILLA._tTotal (PEPZILLA.sol#172) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PEPZILLA._buyMap (PEPZILLA.sol#187) is not in mixedCase
Variable PEPZILLA._maxTxAmount (PEPZILLA.sol#198) is not in mixedCase
Variable PEPZILLA._maxWalletSize (PEPZILLA.sol#199) is not in mixedCase
Variable PEPZILLA._swapTokensAtAmount (PEPZILLA.sol#200) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in PEPZILLA._transfer(address,address,uint256) (PEPZILLA.sol#329-391):
        External calls:
        - sendETHToFee(address(this).balance) (PEPZILLA.sol#364)
                - _marketingAddress.transfer(amount) (PEPZILLA.sol#408)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (PEPZILLA.sol#390)
                - _previousredisFee = _redisFee (PEPZILLA.sol#306)
        - _tokenTransfer(from,to,amount,takeFee) (PEPZILLA.sol#390)
                - _previoustaxFee = _taxFee (PEPZILLA.sol#307)
        - _tokenTransfer(from,to,amount,takeFee) (PEPZILLA.sol#390)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rTeam) (PEPZILLA.sol#471)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (PEPZILLA.sol#461)
```

```
INFO:Detectors:
PEPZILLA.slitherConstructorVariables() (PEPZILLA.sol#159-586) uses literals with too many digits:
        - _maxTxAmount = 2000000000 * 10 ** 9 (PEPZILLA.sol#198)
PEPZILLA.slitherConstructorVariables() (PEPZILLA.sol#159-586) uses literals with too many digits:
        - _maxWalletSize = 3000000000 * 10 ** 9 (PEPZILLA.sol#199)
PEPZILLA.slitherConstructorVariables() (PEPZILLA.sol#159-586) uses literals with too many digits:
        - _swapTokensAtAmount = 1000000000 * 10 ** 9 (PEPZILLA.sol#200)
PEPZILLA.slitherConstructorConstantVariables() (PEPZILLA.sol#159-586) uses literals with too many digits:
        - _tTotal = 100000000000 * 10 ** 9 (PEPZILLA.sol#172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
PEPZILLA._tOwned (PEPZILLA.sol#168) is never used in PEPZILLA (PEPZILLA.sol#159-586)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
PEPZILLA._developmentAddress (PEPZILLA.sol#188) should be constant
PEPZILLA._marketingAddress (PEPZILLA.sol#189) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
PEPZILLA.uniswapV2Pair (PEPZILLA.sol#192) should be immutable
PEPZILLA.uniswapV2Router (PEPZILLA.sol#191) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:PEPZILLA.sol analyzed (7 contracts with 93 detectors), 48 result(s) found
```

Functional Tests
Router (PCS V2):

1- Approve (passed):
https://testnet.bscscan.com/tx/0x43d4bc7221e8e849feb3c4228f40bb59000758e5a1b8538a42ce86fe3e5ac78b

2- Block Bots (passed):
https://testnet.bscscan.com/tx/0x191ca9a44c849347503fd4d4e150257f614291686a53a81135b0f362cc1c31c2

3- Set Fee (passed):
https://testnet.bscscan.com/tx/0xfc87dfb51f663639c1985fc7203135a72dc1023abea4d293b4f726b07b0bf85b

4- Set Max Txn Amount (passed):
https://testnet.bscscan.com/tx/0x03da289b29e8e559fd188da3664041b8bb253fc1b90a5a3957ae0377a0d1568f

5- Set Trading (passed):
https://testnet.bscscan.com/tx/0x1d1504b08046d4359f528fbb264f4d09e71f576599ef5078a7e4b72bf6dfd07f

Ownership Privileges:

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can set trading.
- The owner can set the fees more than 100%.
- The owner can block/unblock address.
- The owner can set max Txn Amount.
- The owner can set max wallet size.
- The owner can exclude multiple accounts from fees.

Findings:
Critical: 0
High: 3
Medium: 0
Low: 2
Informational & Optimizations: 2

Centralization – Buy and Sell fees.

Severity: High
Function: setFee
Status: Open
Overview:
The owner can set the buy and sell fees up to 100%, which is not recommended.

```
function setFee(uint256 redisFeeOnBuy, uint256 redisFeeOnSell, uint256 taxFeeOnBuy,
uint256 taxFeeOnSell) public onlyOwner {
        _redisFeeOnBuy = redisFeeOnBuy;
        _redisFeeOnSell = redisFeeOnSell;
        _taxFeeOnBuy = taxFeeOnBuy;
        _taxFeeOnSell = taxFeeOnSell;
    }
```

Suggestion:
It is recommended that no fees in the contract should be more than 25% of the contract.

Centralization – The owner can Blacklist Wallet.

Severity: High
Function: blockBots
Status: Open
Overview:
The owner can blacklist multiple wallets.

```
function blockBots(address[] memory bots_) public onlyOwner {
        for (uint256 i = 0; i < bots_.length; i++) {
            bots[bots_[i]] = true;
        }
    }
```

Suggestion:
There should be a locking period so that the wallet cannot be locked for an indefinite
Period of time.

Centralization – The owner can lock the token.

Severity: High
Function: setMaxTxnAmount
Status: Open
Overview:
In this changeWalletLimit.

```
function setMaxTxnAmount(uint256 maxTxAmount) public onlyOwner {
        _maxTxAmount = maxTxAmount;
    }
```

Suggestion:
It is recommended that there be a required check for zero address.

Centralization – Missing Events

Severity: Low
Subject: Missing Events
Status: Open
Overview:
They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function setFee(uint256 redisFeeOnBuy, uint256 redisFeeOnSell, uint256 taxFeeOnBuy,
uint256 taxFeeOnSell) public onlyOwner {
        _redisFeeOnBuy = redisFeeOnBuy;
        _redisFeeOnSell = redisFeeOnSell;
        _taxFeeOnBuy = taxFeeOnBuy;
        _taxFeeOnSell = taxFeeOnSell;
    }
function setMaxTxnAmount(uint256 maxTxAmount) public onlyOwner {
        _maxTxAmount = maxTxAmount;
    }
function setMaxWalletSize(uint256 maxWalletSize) public onlyOwner {
        _maxWalletSize = maxWalletSize;
    }

function excludeMultipleAccountsFromFees(address[] calldata accounts, bool ex-
cluded) public onlyOwner {
        for(uint256 i = 0; i < accounts.length; i++) {
            _isExcludedFromFee[accounts[i]] = excluded;
        }
    }
```

Suggestion:
Emit an event for critical changes.

Centralization – Local Variable Shadowing

Severity: Low
Status: Open
Subject: Shadowing Local
Overview:

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

Suggestion:
Rename the local variable that shadows another component.

Optimization

Severity: Informational
Subject: Floating Pragma.
Status: Open
Overview:
It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.9;
```

Suggestion:
Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

Optimization

Severity: **Informational**
Subject: Remove Safe Math
Status: Open
Line: 78-121
Overview:
compiler version above 0.8.0 can control arithmetic overflow/underflow, it is recommended to remove the unwanted code to avoid high gas fees.