Cyberscope

# Audit Report

# Pepenime

May 2023

| | |
|---|---|
| Network | BSC |
| Address | 0x3A6db9D8E127c8aAD48237d4948b652919ee69D4 |
| Audited by | © cyberscope |

# Table of Contents

# Review

| | |
|---|---|
| **Contract Name** | Pepenime |
| **Compiler Version** | v0.8.19+commit.7dd6d404 |
| **Optimization** | 200 runs |
| **Explorer** | https://bscscan.com/address/0x3a6db9d8e127c8aad48237d4948b652919ee69d4 |
| **Address** | 0x3a6db9d8e127c8aad48237d4948b652919ee69d4 |
| **Network** | BSC |
| **Symbol** | Pepenime |
| **Decimals** | 9 |
| **Total Supply** | 100,000,000,000 |

## Audit Updates

| | |
|---|---|
| **Initial Audit** | 23 May 2023 |

## Source Files

| | |
|---|---|
| **Filename** | SHA256 |
| **Pepenime.sol** | d2b8e0c27f4d7bf84bc0ff8f97ef482ce96e3953cc6380cc50e34b6bd6e2a4a8 |

# Findings Breakdown


| | |
|---|---|
| Critical | 0 |
| Medium | 0 |
| Minor / Informative | 10 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 |
| Minor / Informative | 10 | 0 | 0 | 0 |

# Analysis

Critical Medium Minor / Informative Pass

| Severity | Code | Description | Status |
|---|---|---|---|
| ● | ST | Stops Transactions | Passed |
| ● | OCTD | Transfers Contract's Tokens | Passed |
| ● | OTUT | Transfers User's Tokens | Passed |
| ● | ELFM | Exceeds Fees Limit | Passed |
| ● | ULTW | Transfers Liquidity to Team Wallet | Passed |
| ● | MT | Mints Tokens | Passed |
| ● | BT | Burns Tokens | Passed |
| ● | BC | Blacklists Addresses | Passed |

# Diagnostics

Critical Medium Minor / Informative

| Severity | Code | Description | Status |
|---|---|---|---|
| | PTRP | Potential Transfer Revert Propagation | Unresolved |
| | DDP | Decimal Division Precision | Unresolved |
| | RSML | Redundant SafeMath Library | Unresolved |
| | IDI | Immutable Declaration Improvement | Unresolved |
| | L02 | State Variables could be Declared Constant | Unresolved |
| | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| | L05 | Unused State Variable | Unresolved |
| | L16 | Validate Variable Setters | Unresolved |
| | L19 | Stable Compiler Version | Unresolved |
| | L20 | Succeeded Transfer Check | Unresolved |

# PTRP - Potential Transfer Revert Propagation

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol#L297,298 |
| **Status** | Unresolved |

## Description

The contract sends funds to a `marketingWallet` and a `developmentWallet` as part of the transfer flow. These addresses can either be a wallet address or a contract. If the address belongs to a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
_developmentAddress.transfer(amount.div(2));
_marketingAddress.transfer(amount.div(2));
```

## Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be achived by not allowing set contract addresses or by sending the funds in a non-revertable way.

## DDP - Decimal Division Precision

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol#L297 |
| **Status** | Unresolved |

### Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
_developmentAddress.transfer(amount.div(2));
_marketingAddress.transfer(amount.div(2));
```

### Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

# RSML - Redundant SafeMath Library

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol |
| **Status** | Unresolved |

## Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert on underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases the gas consumption unnecessarily.

```
library SafeMath {...}
```

## Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

## IDI - Immutable Declaration Improvement

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol#L173,174 |
| **Status** | Unresolved |

### Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
uniswapV2Router
uniswapV2Pair
```

### Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

## L02 - State Variables could be Declared Constant

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol#L99 |
| **Status** | Unresolved |

### Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
address private _previousOwner
```

### Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

## L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol#L40,138,151,152,153,305,306,311,318,404 |
| **Status** | Unresolved |

### Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function WETH() external pure returns (address);
uint256 private constant _tTotal = 100000000000 * 10**4 * 10**5
string private constant _name = "Pepenime"
string private constant _symbol = "Pepenime"
uint8 private constant _decimals = 9
event tokensRescued(address indexed token, address indexed to, uint
amount);
address _to
address _tokenAddr
uint _amount
event devAddressUpdated(address indexed previous, address indexed adr);
event marketingAddressUpdated(address indexed previous, address indexed
adr);
bool _swapEnabled
```

## Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

## L05 - Unused State Variable

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol#L99,133 |
| **Status** | Unresolved |

### Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
address private _previousOwner
mapping (address => uint256) private _tOwned
```

### Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

## L16 - Validate Variable Setters

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol#L124,314,321 |
| **Status** | Unresolved |

### Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
_owner = newOwner
_developmentAddress = dev
_marketingAddress = markt
```

### Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

## L19 - Stable Compiler Version

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol#L9 |
| **Status** | Unresolved |

### Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.19;
```

### Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

## L20 - Succeeded Transfer Check

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Pepenime.sol#L308 |
| **Status** | Unresolved |

### Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
Token(_tokenAddr).transfer(_to, _amount)
```

### Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

# Functions Analysis

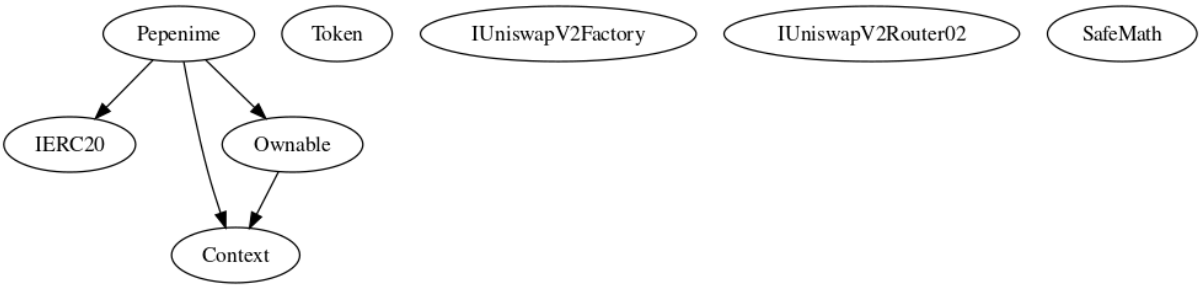
| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **IERC20** | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| **Token** | Interface | | | |
| | transferFrom | External | ✓ | - |
| | transfer | External | ✓ | - |
| | | | | |
| **IUniswapV2Factory** | Interface | | | |
| | createPair | External | ✓ | - |
| | | | | |
| **IUniswapV2Router02** | Interface | | | |
| | swapExactTokensForETHSupportingFeeOnTransferTokens | External | ✓ | - |

| | | | | |
|---|---|---|---|---|
| | factory | External | | - |
| | WETH | External | | - |
| | addLiquidityETH | External | Payable | - |
| | | | | |
| **Context** | Implementation | | | |
| | _msgSender | Internal | | |
| | | | | |
| **SafeMath** | Library | | | |
| | add | Internal | | |
| | sub | Internal | | |
| | sub | Internal | | |
| | mul | Internal | | |
| | div | Internal | | |
| | div | Internal | | |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | | Public | ✓ | - |
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | | | | |
| **Pepenime** | Implementation | Context, IERC20, Ownable | | |
| | | Public | ✓ | - |

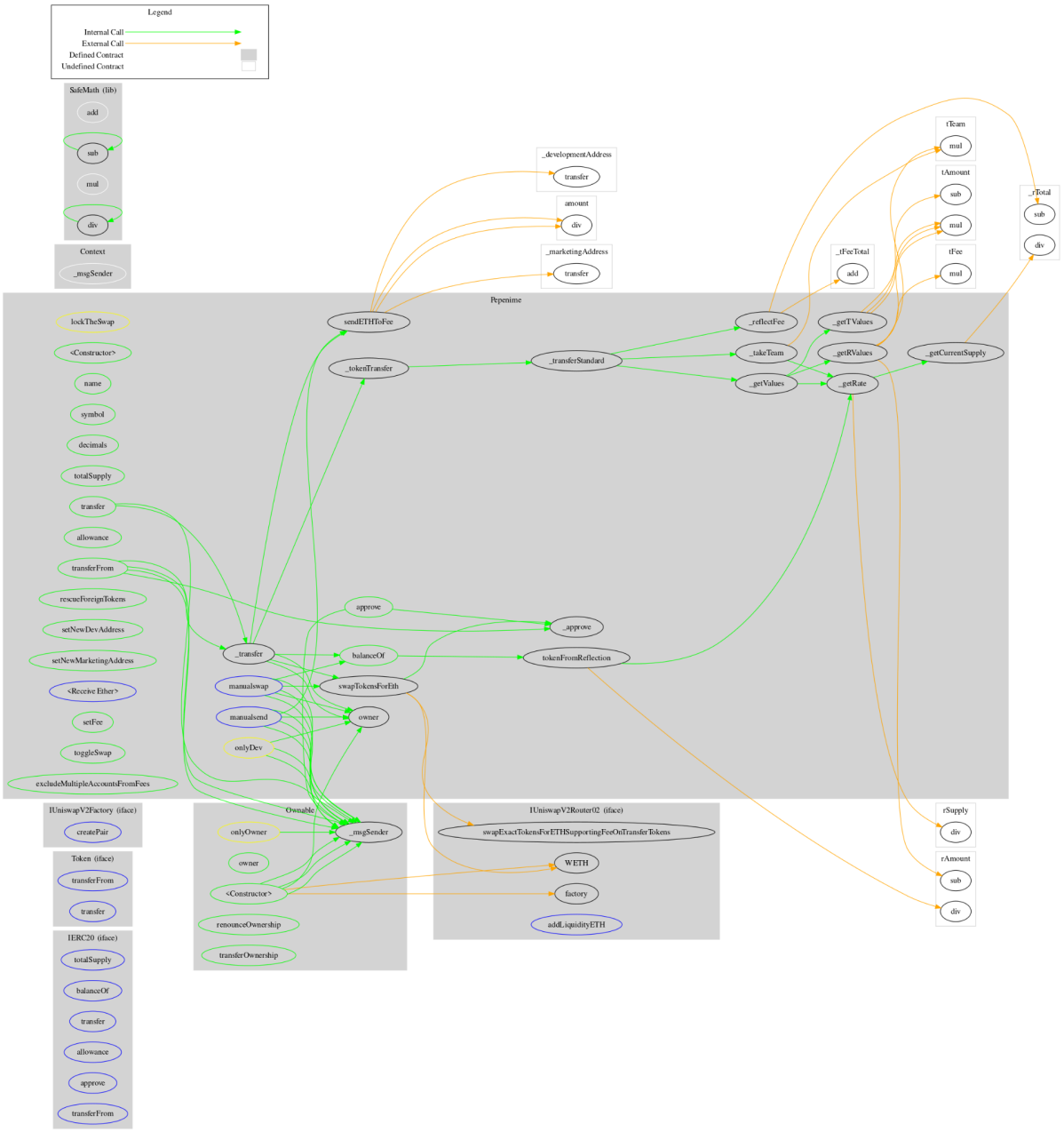
| | | | | |
|---|---|---|---|---|
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | Public | | - |
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | tokenFromReflection | Private | | |
| | _approve | Private | ✓ | |
| | _transfer | Private | ✓ | |
| | swapTokensForEth | Private | ✓ | lockTheSwap |
| | sendETHToFee | Private | ✓ | |
| | _tokenTransfer | Private | ✓ | |
| | rescueForeignTokens | Public | ✓ | onlyDev |
| | setNewDevAddress | Public | ✓ | onlyDev |
| | setNewMarketingAddress | Public | ✓ | onlyDev |
| | _transferStandard | Private | ✓ | |
| | _takeTeam | Private | ✓ | |
| | _reflectFee | Private | ✓ | |
| | | External | Payable | - |
| | _getValues | Private | | |

| | | | | |
|---|---|---|---|---|
| | _getTValues | Private | | |
| | _getRValues | Private | | |
| | _getRate | Private | | |
| | _getCurrentSupply | Private | | |
| | manualswap | External | ✓ | - |
| | manualsend | External | ✓ | - |
| | setFee | Public | ✓ | onlyDev |
| | toggleSwap | Public | ✓ | onlyDev |
| | excludeMultipleAccountsFromFees | Public | ✓ | onlyOwner |

# Inheritance Graph



# Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SafeMath (lib)
add
sub
mul
div
Context
_msgSender
_developmentAddress
transfer
amount
div
_marketingAddress
transfer
Pepenime
lockTheSwap
<Constructor>
name
symbol
decimals
totalSupply
transfer
allowance
transferFrom
rescueForeignTokens
setNewDevAddress
setNewMarketingAddress
<Receive Ether>
setFee
toggleSwap
excludeMultipleAccountsFromFees
sendETHToFee
_tokenTransfer
_transfer
manualswap
manualsend
onlyDev
approve
_approve
balanceOf
swapTokensForEth
owner
tokenFromReflection
_transferStandard
_reflectFee
_takeTeam
_getValues
_getTValues
_getRValues
_getRate
_getCurrentSupply
tTeam
mul
tAmount
sub
mul
_rTotal
sub
div
_tFeeTotal
add
tFee
mul
rSupply
div
rAmount
sub
div
IUniswapV2Factory (iface)
createPair
Token (iface)
transferFrom
transfer
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
onlyOwner
owner
<Constructor>
renounceOwnership
transferOwnership
_msgSender
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
addLiquidityETH

# Summary

Pepenime contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Pepenime is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 4% buy and sell fees.

# Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

# About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

**The Cyberscope team**

https://www.cyberscope.io