# Audit Report

# Cicca Defi

October 2023

| | |
|---|---|
| Network | BSC |
| Address | 0xa4320f7756bdbf4796b77bea6f2a3432c60f8456 |
| Audited by | © cyberscope |

# Analysis

Critical Medium Minor / Informative Pass

| Severity | Code | Description | Status |
|---|---|---|---|
| Pass | ST | Stops Transactions | Passed |
| Pass | OTUT | Transfers User's Tokens | Passed |
| Critical | ELFM | Exceeds Fees Limit | Unresolved |
| Pass | MT | Mints Tokens | Passed |
| Pass | BT | Burns Tokens | Passed |
| Pass | BC | Blacklists Addresses | Passed |

# Diagnostics

● Critical ● Medium ● Minor / Informative

| Severity | Code | Description | Status |
|---|---|---|---|
| Medium | FRV | Fee Restoration Vulnerability | Unresolved |
| Minor / Informative | IFM | Inconsistent Fee Mechanism | Unresolved |
| Minor / Informative | MEM | Misleading Error Messages | Unresolved |
| Minor / Informative | AOI | Arithmetic Operations Inconsistency | Unresolved |
| Minor / Informative | RRS | Redundant Require Statement | Unresolved |
| Minor / Informative | FRV | Fee Restoration Vulnerability | Unresolved |
| Minor / Informative | EPC | Existing Pair Creation | Unresolved |
| Minor / Informative | RSW | Redundant Storage Writes | Unresolved |
| Minor / Informative | RES | Redundant Event Statement | Unresolved |
| Minor / Informative | RSML | Redundant SafeMath Library | Unresolved |
| Minor / Informative | IDI | Immutable Declaration Improvement | Unresolved |
| Minor / Informative | L02 | State Variables could be Declared Constant | Unresolved |
| Minor / Informative | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| Minor / Informative | L07 | Missing Events Arithmetic | Unresolved |

| | | | |
|---|---|---|---|
| ● | L09 | Dead Code Elimination | Unresolved |
| ● | L17 | Usage of Solidity Assembly | Unresolved |

# Table of Contents


| | |
|---|---|
| **Analysis** | **1** |
| **Diagnostics** | **2** |
| **Table of Contents** | **4** |
| **Review** | **6** |
| Audit Updates | 6 |
| Source Files | 6 |
| **Findings Breakdown** | **7** |
| ELFM - Exceeds Fees Limit | 8 |
| Description | 8 |
| Recommendation | 8 |
| FRV - Fee Restoration Vulnerability | 10 |
| Description | 10 |
| Recommendation | 12 |
| IFM - Inconsistent Fee Mechanism | 13 |
| Description | 13 |
| Recommendation | 14 |
| MEM - Misleading Error Messages | 15 |
| Description | 15 |
| Recommendation | 15 |
| AOI - Arithmetic Operations Inconsistency | 17 |
| Description | 17 |
| Recommendation | 17 |
| RRS - Redundant Require Statement | 18 |
| Description | 18 |
| Recommendation | 18 |
| EPC - Existing Pair Creation | 19 |
| Description | 19 |
| Recommendation | 19 |
| RSW - Redundant Storage Writes | 20 |
| Description | 20 |
| Recommendation | 21 |
| RES - Redundant Event Statement | 23 |
| Description | 23 |
| Recommendation | 23 |
| RSML - Redundant SafeMath Library | 24 |
| Description | 24 |
| Recommendation | 24 |
| IDI - Immutable Declaration Improvement | 25 |
| Description | 25 |

Recommendation 25
L02 - State Variables could be Declared Constant 26
Description 26
Recommendation 26
L04 - Conformance to Solidity Naming Conventions 27
Description 27
Recommendation 28
L07 - Missing Events Arithmetic 29
Description 29
Recommendation 29
L09 - Dead Code Elimination 30
Description 30
Recommendation 30
L17 - Usage of Solidity Assembly 31
Description 31
Recommendation 31
**Functions Analysis 32**
**Inheritance Graph 40**
**Flow Graph 41**
**Summary 42**
**Disclaimer 43**
**About Cyberscope 44**

# Review

| | |
|---|---|
| **Contract Name** | CiccaDefi |
| **Compiler Version** | v0.8.19+commit.7dd6d404 |
| **Optimization** | 200 runs |
| **Explorer** | https://bscscan.com/address/0xa4320f7756bdbf4796b77bea6f2a3432c60f8456 |
| **Address** | 0xa4320f7756bdbf4796b77bea6f2a3432c60f8456 |
| **Network** | BSC |
| **Symbol** | CICCA |
| **Decimals** | 18 |
| **Total Supply** | 10,000,000,000 |

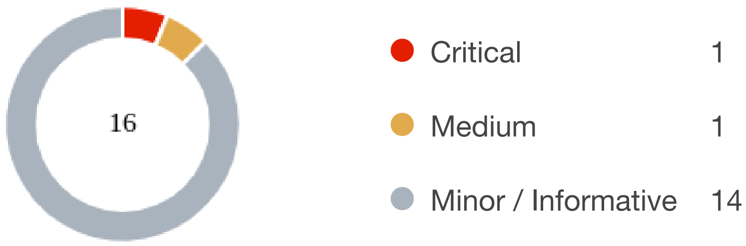
## Audit Updates

| | |
|---|---|
| **Initial Audit** | 16 Oct 2023 |

## Source Files

| | |
|---|---|
| **Filename** | SHA256 |
| **CiccaDefi.sol** | ff169810b5b4b5f88a4c4e54b67dce1a2293449008bb1ba7a87d9c4c4ccb1575 |

# Findings Breakdown




| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| Critical | 1 | 0 | 0 | 0 |
| Medium | 1 | 0 | 0 | 0 |
| Minor / Informative | 14 | 0 | 0 | 0 |

## ELFM - Exceeds Fees Limit

| | |
|---|---|
| **Criticality** | Critical |
| **Location** | CiccaDefi.sol#L1139 |
| **Status** | Unresolved |

### Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setTaxFeePercent` function with a high percentage value.

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner()
{
    _taxFee = taxFee;
}

function setLiquidityFeePercent(uint256 liquidityFee) external
onlyOwner() {
    _liquidityFee = liquidityFee;
}

function setStakingFeePercent(uint256 stakingFee) external
onlyOwner() {
    _stakingFee = stakingFee;
}
```

### Recommendation

The contract could embody a check for the maximum acceptable value. 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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.

- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

## FRV - Fee Restoration Vulnerability

| | |
|---|---|
| **Criticality** | Medium |
| **Location** | CiccaDefi.sol#L944,956,1094 |
| **Status** | Unresolved |

### Description

The contract demonstrates a potential vulnerability upon removing and restoring the fees. This vulnerability can occur when the fees have been set to zero. During a transaction, if the fees have been set to zero, then both remove fees and restore fees functions will be executed. The remove fees function is executed to temporarily remove the fees, ensuring the sender is not taxed during the transfer. However, the function prematurely returns without setting the variables that hold the previous fee values.

As a result, when the subsequent restore fees function is called after the transfer, it restores the fees to their previous values. However, since the previous fee values were not properly set to zero, there is a risk that the fees will retain their non-zero values from before the fees were removed. This can lead to unintended consequences, potentially causing incorrect fee calculations or unexpected behavior within the contract.

```
    function removeAllFee() private {
        if(_taxFee == 0 && _liquidityFee == 0 && _stakingFee==0) return;

        _previousTaxFee = _taxFee;
        _previousLiquidityFee = _liquidityFee;
        _previousstakingFee = _stakingFee;

        _taxFee = 0;
        _liquidityFee = 0;
        _stakingFee = 0;
    }

    function restoreAllFee() private {
       _taxFee = _previousTaxFee;
       _liquidityFee = _previousLiquidityFee;
       _stakingFee = _previousstakingFee;
    }

    function _tokenTransfer(address sender, address recipient, uint256
amount) private {
        if(_isExcludedFromFee[sender] || _isExcludedFromFee[recipient]){
            removeAllFee();
        }
        ...
        uint256 stakingAmt = amount.mul(_stakingFee).div(100);

        if (_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferFromExcluded(sender, recipient,
(amount.sub(stakingAmt)));
       ...
        //Temporarily remove fees to transfer to staking wallet
        _taxFee = 0;
        _liquidityFee = 0;

        _transferStandard(sender, stakingWallet, stakingAmt);

        //Restore tax and liquidity fees
        _taxFee = _previousTaxFee;
        _liquidityFee = _previousLiquidityFee;


        if(_isExcludedFromFee[sender] || _isExcludedFromFee[recipient])
            restoreAllFee();
    }
```

## Recommendation

The team is advised to modify the remove fees function to ensure that the previous fee values are correctly set to zero, regardless of their initial values. A recommended approach would be to remove the early return when both fees are zero.

## IFM - Inconsistent Fee Mechanism

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L932,1069 |
| **Status** | Unresolved |

### Description

The contract calculates a `stakingAmt` variable, which represents a percentage of the transferred amount. This percentage is deducted from the original amount, and the tax is then applied to the remaining amount. As a result, the fees are not calculated proportionally. Instead, a percentage amount is taken as fees, and an additional tax amount is applied to the remaining amount. This approach can lead to unintended consequences and may not align with the expected behavior of a standard tax fee system.

```
function _tokenTransfer(address sender, address recipient, uint256
amount) private {
        if(_isExcludedFromFee[sender] || _isExcludedFromFee[recipient]){
            removeAllFee();
        }

        //Calculate staking amount
        uint256 stakingAmt = amount.mul(_stakingFee).div(100);

        if (_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferFromExcluded(sender, recipient,
(amount.sub(stakingAmt)));
        } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
            _transferToExcluded(sender, recipient,
(amount.sub(stakingAmt)));
        } else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferStandard(sender, recipient,
(amount.sub(stakingAmt)));
        } else if (_isExcluded[sender] && _isExcluded[recipient]) {
            _transferBothExcluded(sender, recipient,
(amount.sub(stakingAmt)));
        } else {
            _transferStandard(sender, recipient,
(amount.sub(stakingAmt)));
        }

        ...
    }

    function calculateTaxFee(uint256 _amount) private view returns
(uint256) {
        return _amount.mul(_taxFee).div(
            10**2
        );
    }
```

## Recommendation

It is recommended to adopt a percentage-wise method for calculating the tax fees. Given that the intended functionality is to compute a tax fee, the contract could implement a standard percentage tax fee. This ensures that the tax is applied uniformly to the entire amount, providing clarity and predictability to the users of the contract.

## MEM - Misleading Error Messages

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L477 |
| **Status** | Unresolved |

### Description

The contract is using misleading error messages. These error messages do not accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

The contract is designed to allow the owner to lock the contract for a specified amount of time using the `lock` function. This function sets the `_lockTime` variable by adding the provided time parameter to the current `block.timestamp`. However, while the unlock function's error message suggests that the contract is locked for 7 days, there is no actual check in the `lock` function to ensure that the value of the time parameter, and consequently the `_lockTime` variable, is equal to or greater than 7 days. As a result the `_lockTime` could be set to any value, even lower than the 7 days period.

```
    function lock(uint256 time) public virtual onlyOwner {
        _previousOwner = _owner;
        _owner = address(0);
        _lockTime = block.timestamp + time;
        emit OwnershipTransferred(_owner, address(0));
    }

    function unlock() public virtual {
        require(_previousOwner == msg.sender, "You don't have
permission to unlock");
        require(block.timestamp > _lockTime , "Contract is
locked until 7 days");
        emit OwnershipTransferred(_owner, _previousOwner);
        _owner = _previousOwner;
    }
```

### Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract. It is recommended that if the intended functionality is to set the `_lockTime` variable to a minimum of 7 days, then the `lock` function should embody a check to prevent the setting of the `_lockTime` variable to a value lower than 7 days. This can be achieved by adding a require statement in the `lock` function to validate the time parameter.

# AOI - Arithmetic Operations Inconsistency

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L470,883 |
| **Status** | Unresolved |

## Description

The contract uses both the SafeMath library and native arithmetic operations. The SafeMath library is commonly used to mitigate vulnerabilities related to integer overflow and underflow issues. However, it was observed that the contract also employs native arithmetic operators (such as +, -, *, /) in certain sections of the code.

The combination of SafeMath library and native arithmetic operations can introduce inconsistencies and undermine the intended safety measures. This discrepancy creates an inconsistency in the contract's arithmetic operations, increasing the risk of unintended consequences such as inconsistency in error handling, or unexpected behavior.

```
_lockTime = block.timestamp + time;
...
_tFeeTotal = _tFeeTotal.add(tFee);
```

## Recommendation

To address this finding and ensure consistency in arithmetic operations, it is recommended to standardize the usage of arithmetic operations throughout the contract. The contract should be modified to either exclusively use SafeMath library functions or entirely rely on native arithmetic operations, depending on the specific requirements and design considerations. This consistency will help maintain the contract's integrity and mitigate potential vulnerabilities arising from inconsistent arithmetic operations.

## RRS - Redundant Require Statement

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L118 |
| **Status** | Unresolved |

### Description

The contract utilizes a `require` statement within the `add` function aiming to prevent overflow errors. This function is designed based on the SafeMath library's principles. In Solidity version 0.8.0 and later, arithmetic operations revert on overflow and underflow, making the overflow check within the function redundant. This redundancy could lead to extra gas costs and increased complexity without providing additional security.

```
function add(uint256 a, uint256 b) internal pure returns
(uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");
        return c;
}
```

### Recommendation

It is recommended to remove the `require` statement from the add function since the contract is using a Solidity pragma version equal to or greater than 0.8.0. By doing so, the contract will leverage the built-in overflow and underflow checks provided by the Solidity language itself, simplifying the code and reducing gas consumption. This change will uphold the contract's integrity in handling arithmetic operations while optimizing for efficiency and cost-effectiveness.

## EPC - Existing Pair Creation

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L1151 |
| **Status** | Unresolved |

### Description

The contract contains the `setRouterAddress` function that does not handle the scenario where a pair already exists prior to its execution. If a pair for the given tokens has already been established, the `createPair` function will revert and not proceed with the creation of a new pair. As a result, if a pair has been previously set up before the function is invoked, the contract will encounter an error when trying to call the `createPair` function. This will prevent the successful execution, essentially leading the function to revert.

```
    function setRouterAddress(address newRouter) public onlyOwner()
{
        IUniswapV2Router02 _newPancakeRouter =
IUniswapV2Router02(newRouter);
        uniswapV2Pair =
IUniswapV2Factory(_newPancakeRouter.factory()).createPair(address(t
his), _newPancakeRouter.WETH());
        uniswapV2Router = _newPancakeRouter;
    }
```

### Recommendation

To mitigate the risks associated with attempting to create an already existing pair, it is recommended to implement a check to determine whether the pair already exists before proceeding to create a new pair. This can be achieved by utilizing the getPair function of the Factory contract to retrieve the address of the pair contract for the specified tokens. If the address returned by the getPair function is the zero address, it indicates that the pair does not exist, and the contract can proceed with the createPair function. Conversely, if a non-zero address is returned, it indicates that the pair already exists, and the createPair function will revert.

## RSW - Redundant Storage Writes

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L1127 |
| **Status** | Unresolved |

### Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
    function excludeFromFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = true;
    }

    function includeInFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = false;
    }

    function setStakingWallet(address newWallet) external
onlyOwner() {
        stakingWallet = newWallet;
    }

    function setTaxFeePercent(uint256 taxFee) external onlyOwner()
{
        _taxFee = taxFee;
    }

    function setLiquidityFeePercent(uint256 liquidityFee) external
onlyOwner() {
        _liquidityFee = liquidityFee;
    }

    function setStakingFeePercent(uint256 stakingFee) external
onlyOwner() {
        _stakingFee = stakingFee;
    }

    function setRouterAddress(address newRouter) public onlyOwner()
{
        IUniswapV2Router02 _newPancakeRouter =
IUniswapV2Router02(newRouter);
        uniswapV2Pair =
IUniswapV2Factory(_newPancakeRouter.factory()).createPair(address(t
his), _newPancakeRouter.WETH());
        uniswapV2Router = _newPancakeRouter;
    }

    function setSwapAndLiquifyEnabled(bool _enabled) public
onlyOwner {
        swapAndLiquifyEnabled = _enabled;
        emit SwapAndLiquifyEnabledUpdated(_enabled);
    }
```

## Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

# RES - Redundant Event Statement

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L726 |
| **Status** | Unresolved |

## Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The `MinTokensBeforeSwapUpdated` event statement is not used in the contract's implemantation.

```
event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

## Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. It is recommend removing the unused event statement from the contract..

## RSML - Redundant SafeMath Library

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.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 to 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 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** | CiccaDefi.sol#L739 |
| **Status** | Unresolved |

### Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
_routerAddress
```

### 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** | CiccaDefi.sol#L693,697,698,699,720 |
| **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.

```
uint256 private _tTotal = 10000000000 * 10**18; //10 billion
total token
string private _name = "Cicca DeFi";
string private _symbol = "CICCA";
uint8 private _decimals = 18;
uint256 private numTokensSellToAddToLiquidity = 100000 *
10**18;
```

### 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** | CiccaDefi.sol#L510,511,527,546,701,704,707,715,928,934,1153 |
| **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 DOMAIN_SEPARATOR() external view returns (bytes32);
function PERMIT_TYPEHASH() external pure returns (bytes32);
function MINIMUM_LIQUIDITY() external pure returns (uint);
function WETH() external pure returns (address);
uint256 public _taxFee = 1;
uint256 public _liquidityFee = 1;
uint256 public _stakingFee = 1;
address _routerAddress;
function calculateTaxFee(uint256 _amount) private view returns
(uint256)
function calculateLiquidityFee(uint256 _amount) private view
returns (uint256)
...
```

## 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.

# L07 - Missing Events Arithmetic

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L1136,1140,1144,1150 |
| **Status** | Unresolved |

## Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
_taxFee = taxFee;
_liquidityFee = liquidityFee;
_stakingFee = stakingFee;
uniswapV2Router = _newPancakeRouter;
```

## Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

## L09 - Dead Code Elimination

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L303 |
| **Status** | Unresolved |

### Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
}
    function sendValue(address payable recipient, uint256
amount) internal {
        require(address(this).balance >= amount, "Address:
insufficient balance");

        // solhint-disable-next-line avoid-low-level-calls,
avoid-call-value
        (bool success, ) = recipient.call{ value: amount }("");
        require(success, "Address: unable to send value,
recipient may have reverted");
    }
...
```

### Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

## L17 - Usage of Solidity Assembly

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | CiccaDefi.sol#L283,382 |
| **Status** | Unresolved |

### Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly { codehash := extcodehash(account) }
assembly {
    let returndata_size := mload(returndata)
    revert(add(32, returndata), returndata_size)
    }
```

### Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

# Functions Analysis

| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **IERC20** | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| **SafeMath** | Library | | | |
| | add | Internal | | |
| | sub | Internal | | |
| | sub | Internal | | |
| | mul | Internal | | |
| | div | Internal | | |
| | div | Internal | | |
| | mod | Internal | | |
| | mod | Internal | | |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **Context** | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **Address** | Library | | | |
| | isContract | Internal | | |
| | sendValue | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | _functionCallWithValue | Private | ✓ | |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | | Public | ✓ | - |
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | geUnlockTime | Public | | - |
| | lock | Public | ✓ | onlyOwner |
| | unlock | Public | ✓ | - |
| | | | | |
| **IUniswapV2Factory** | Interface | | | |

| | | | | |
|---|---|---|---|---|
| | feeTo | External | | - |
| | feeToSetter | External | | - |
| | getPair | External | | - |
| | allPairs | External | | - |
| | allPairsLength | External | | - |
| | createPair | External | ✓ | - |
| | setFeeTo | External | ✓ | - |
| | setFeeToSetter | External | ✓ | - |
| | | | | |
| **IUniswapV2Pair** | Interface | | | |
| | name | External | | - |
| | symbol | External | | - |
| | decimals | External | | - |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transfer | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | DOMAIN_SEPARATOR | External | | - |
| | PERMIT_TYPEHASH | External | | - |
| | nonces | External | | - |
| | permit | External | ✓ | - |

| | | | | |
|---|---|---|---|---|
| | MINIMUM_LIQUIDITY | External | | - |
| | factory | External | | - |
| | token0 | External | | - |
| | token1 | External | | - |
| | getReserves | External | | - |
| | price0CumulativeLast | External | | - |
| | price1CumulativeLast | External | | - |
| | kLast | External | | - |
| | burn | External | ✓ | - |
| | swap | External | ✓ | - |
| | skim | External | ✓ | - |
| | sync | External | ✓ | - |
| | initialize | External | ✓ | - |
| | | | | |
| **IUniswapV2Router01** | Interface | | | |
| | factory | External | | - |
| | WETH | External | | - |
| | addLiquidity | External | ✓ | - |
| | addLiquidityETH | External | Payable | - |
| | removeLiquidity | External | ✓ | - |
| | removeLiquidityETH | External | ✓ | - |
| | removeLiquidityWithPermit | External | ✓ | - |
| | removeLiquidityETHWithPermit | External | ✓ | - |

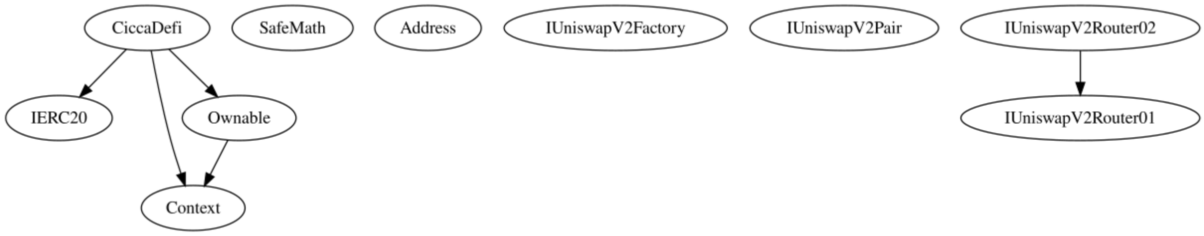
| | | | | |
|---|---|---|---|---|
| | swapExactTokensForTokens | External | ✓ | - |
| | swapTokensForExactTokens | External | ✓ | - |
| | swapExactETHForTokens | External | Payable | - |
| | swapTokensForExactETH | External | ✓ | - |
| | swapExactTokensForETH | External | ✓ | - |
| | swapETHForExactTokens | External | Payable | - |
| | quote | External | | - |
| | getAmountOut | External | | - |
| | getAmountIn | External | | - |
| | getAmountsOut | External | | - |
| | getAmountsIn | External | | - |
| | | | | |
| **IUniswapV2Router02** | Interface | IUniswapV2Router01 | | |
| | removeLiquidityETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactETHForTokensSupportingFeeOnTransferTokens | External | Payable | - |
| | swapExactTokensForETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | | | | |
| **CiccaDefi** | Implementation | Context, IERC20, Ownable | | |
| | | Public | ✓ | - |

| | | | | |
|---|---|---|---|---|
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | Public | | - |
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | isExcludedFromReward | Public | | - |
| | totalFees | Public | | - |
| | deliver | Public | ✓ | - |
| | reflectionFromToken | Public | | - |
| | tokenFromReflection | Public | | - |
| | excludeFromReward | Public | ✓ | onlyOwner |
| | includeInReward | External | ✓ | onlyOwner |
| | _transferBothExcluded | Private | ✓ | |
| | | External | Payable | - |
| | _reflectFee | Private | ✓ | |
| | _getValues | Private | | |
| | _getTValues | Private | | |

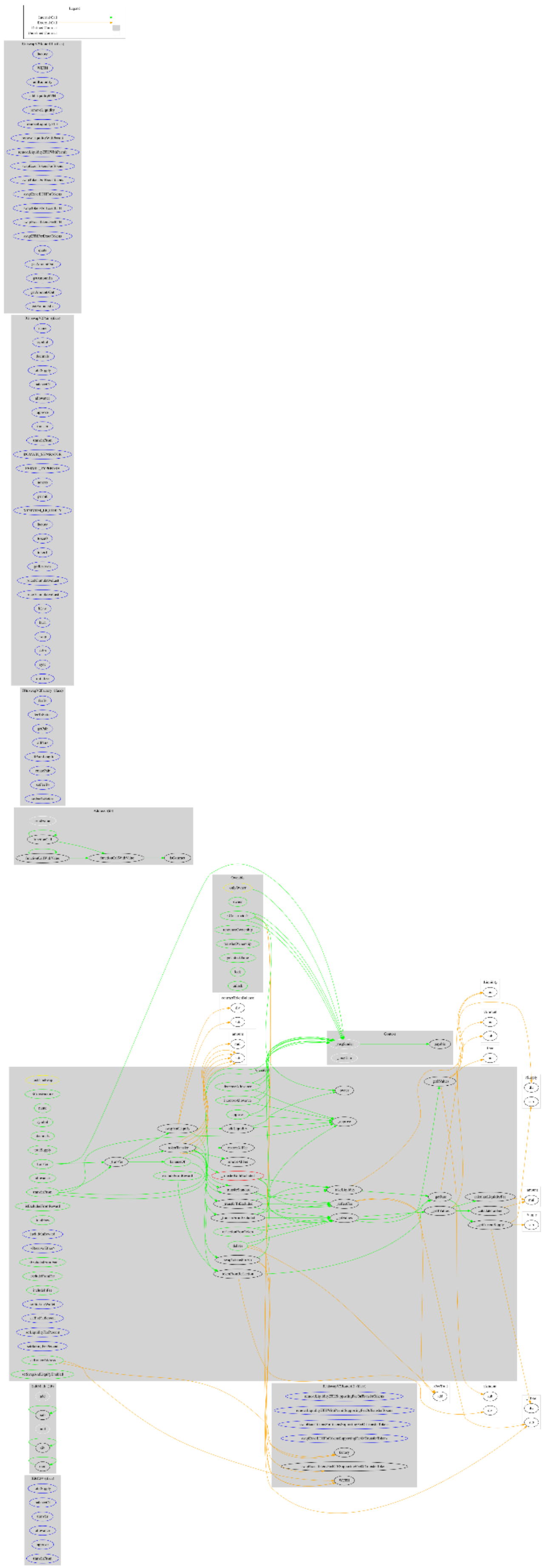
| | | | | |
|---|---|---|---|---|
| | _getRValues | Private | | |
| | _getRate | Private | | |
| | _getCurrentSupply | Private | | |
| | _takeLiquidity | Private | ✓ | |
| | calculateTaxFee | Private | | |
| | calculateLiquidityFee | Private | | |
| | removeAllFee | Private | ✓ | |
| | restoreAllFee | Private | ✓ | |
| | isExcludedFromFee | Public | | - |
| | _approve | Private | ✓ | |
| | _transfer | Private | ✓ | |
| | swapAndLiquify | Private | ✓ | lockTheSwap |
| | swapTokensForEth | Private | ✓ | |
| | addLiquidity | Private | ✓ | |
| | _tokenTransfer | Private | ✓ | |
| | _transferStandard | Private | ✓ | |
| | _transferToExcluded | Private | ✓ | |
| | _transferFromExcluded | Private | ✓ | |
| | excludeFromFee | Public | ✓ | onlyOwner |
| | includeInFee | Public | ✓ | onlyOwner |
| | setStakingWallet | External | ✓ | onlyOwner |
| | setTaxFeePercent | External | ✓ | onlyOwner |
| | setLiquidityFeePercent | External | ✓ | onlyOwner |

| | | | | |
|---|---|---|---|---|
| | setStakingFeePercent | External | ✓ | onlyOwner |
| | setRouterAddress | Public | ✓ | onlyOwner |
| | setSwapAndLiquifyEnabled | Public | ✓ | onlyOwner |

# Inheritance Graph



# Flow Graph



# Summary

Cicca Defi contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like manipulate the fees. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

# 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