# Cyberscope

## Audit Report
# Chris World Asset

December 2023

# Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

# Diagnostics

● Critical  ● Medium  ● Minor / Informative

| Severity | Code | Description | Status |
|---|---|---|---|
| ● | PRV | Potential Router Vulnerability | Unresolved |
| ● | RED | Redundant Event Declaration | Unresolved |
| ● | EPC | Existing Pair Creation | Unresolved |
| ● | PVC | Price Volatility Concern | Unresolved |
| ● | MEE | Missing Events Emission | Unresolved |
| ● | RSW | Redundant Storage Writes | Unresolved |
| ● | RSML | Redundant SafeMath Library | Unresolved |
| ● | IDI | Immutable Declaration Improvement | Unresolved |
| ● | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| ● | L07 | Missing Events Arithmetic | Unresolved |
| ● | L09 | Dead Code Elimination | Unresolved |
| ● | L13 | Divide before Multiply Operation | Unresolved |
| ● | L16 | Validate Variable Setters | Unresolved |
| ● | L17 | Usage of Solidity Assembly | Unresolved |

| | L19 | Stable Compiler Version | Unresolved |
|---|---|---|---|

# Table of Contents

# Review

| | |
|---|---|
| **Contract Name** | CoinToken |
| **Compiler Version** | v0.8.4+commit.c7e474f2 |
| **Optimization** | 200 runs |
| **Testing Deploy** | https://mumbai.polygonscan.com/address/0x66b5036C6F43ef5Ac2fF550fE5459b3F1219f0c1 |
| **Explorer** | https://sepolia.etherscan.io/address/0x98b7b6b90b20883a39dd0a7f645652c7b3389351 |
| **Address** | 0x98b7b6b90b20883a39dd0a7f645652c7b3389351 |
| **Network** | ETH Sepolia |
| **Symbol** | CWA |
| **Decimals** | 18 |
| **Total Supply** | 600,000,000 |

## Audit Updates

| | |
|---|---|
| **Initial Audit** | 11 Dec 2023 |

## Source Files

| Filename | SHA256 |
|---|---|
| **contracts/CoinToken.sol** | e077b434f64947aed73a7900b0c9ec0df0abf47d727a4a31247a7d0e0e748d4c |

# Findings Breakdown



| | Critical | 2 |
|---|---|---|
| | Medium | 0 |
| | Minor / Informative | 15 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| Critical | 2 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 |
| Minor / Informative | 15 | 0 | 0 | 0 |

## ST - Stops Transactions

| | |
|---|---|
| **Criticality** | Critical |
| **Location** | contracts/CoinToken.sol#L793 |
| **Status** | Unresolved |

## Description

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by setting the `_maxTxAmount` to zero.

```
if(from != owner() && to != owner())
    require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
```

Additionally, the contract owner has the authority to stop the sales for all users excluding the owner, as described in detail in sections PRV and PVC. As a result, the contract might operate as a honeypot.

## Recommendation

The contract could embody a check for not allowing setting the `_maxTxAmount` less than a reasonable amount. A suggested implementation could check that the minimum amount should be more than a fixed percentage of the total supply. 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.

# ELFM - Exceeds Fees Limit

| | |
|---|---|
| **Criticality** | Critical |
| **Location** | contracts/CoinToken.sol#L648,652,656 |
| **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 either of the following functions with a high percentage value:

- `setTaxFeePercent` .
- `setDevFeePercent` .
- `setLiquidityFeePercent` .

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

function setDevFeePercent(uint256 devFee) external onlyOwner() {
    _devFee = devFee;
}

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

## 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.

# PRV - Potential Router Vulnerability

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/CoinToken.sol#L910 |
| **Status** | Unresolved |

## Description

The contract includes a method that allows the owner to modify the router address and create a new pair. While this feature provides flexibility, it introduces a security threat. The owner could set the router address to any contract that implements the router's interface, potentially containing malicious code. In the event of a transaction triggering the swap functionality with such a malicious contract as the router, the transaction may revert due to potential manipulations in the swap function.

```
function setRouterAddress(address newRouter) external onlyOwner {
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(newRouter);
    uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),
_uniswapV2Router.WETH());
    uniswapV2Router = _uniswapV2Router;
}
```

## 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.

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.

# RED - Redundant Event Declaration

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/CoinToken.sol#L470 |
| **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 contract declares certain events in its code. However, these events are not emitted within the contract's functions. As a result, these declared events are redundant and serve no purpose within the contract's current implementation.

```solidity
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.

A recommended approach would be to either remove the declared events that are not being emitted or to incorporate the necessary emit statements within the contract's functions to actually emit these events when relevant actions occur.

# EPC - Existing Pair Creation

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/CoinToken.sol#L910 |
| **Status** | Unresolved |

## Description

The contract contains a 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) external onlyOwner {
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(newRouter);
    uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),
_uniswapV2Router.WETH());
    uniswapV2Router = _uniswapV2Router;
}
```

## 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.

# PVC - Price Volatility Concern

| Criticality | Minor / Informative |
| --- | --- |
| Location | contracts/CoinToken.sol#L916 |
| Status | Unresolved |

## Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `numTokensSellToAddToLiquidity` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH. This poses a significant risk, especially if the swap amount exceeds the router's slippage threshold. Such a scenario could result in the failure of the sale transaction.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function setNumTokensSellToAddToLiquidity(uint256 amountToUpdate) external
onlyOwner {
    numTokensSellToAddToLiquidity = amountToUpdate;
}
```

## Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

# MEE - Missing Events Emission

| Criticality | Minor / Informative |
|---|---|
| Location | contracts/CoinToken.sol#L641,645,665,725,733,913 |
| Status | Unresolved |

## Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
_isExcludedFromFee[account] = true;
_isExcludedFromFee[account] = false;
_devWalletAddress = _addr;
_rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity);
_rOwned[_devWalletAddress] = _rOwned[_devWalletAddress].add(rDev);
uniswapV2Router = _uniswapV2Router;
```

## Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

## RSW - Redundant Storage Writes

| Criticality | Minor / Informative |
|---|---|
| Location | contracts/CoinToken.sol#L641,645,670 |
| 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.

```
_isExcludedFromFee[account] = true;
_isExcludedFromFee[account] = false;
swapAndLiquifyEnabled = _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.

## RSML - Redundant SafeMath Library

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/CoinToken.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.

```solidity
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** | contracts/CoinToken.sol#L487,488 |
| **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.

```
_decimals
_tTotal
```

## 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.

## L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/CoinToken.sol#L202,204,275,276,290,308,450,458,460,462,468,664,669,738,744,750 |
| **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.

```
address public _owner
uint256 public _lockTime
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);
address public _devWalletAddress
uint256 public _taxFee
uint256 public _devFee
uint256 public _liquidityFee
uint256 public _maxTxAmount
address _addr
bool _enabled
uint256 _amount
```

## 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** | contracts/CoinToken.sol#L649,653,657,661,917 |
| **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
_devFee = devFee
_liquidityFee = liquidityFee
_maxTxAmount = maxTxPercent  * 10 ** _decimals
numTokensSellToAddToLiquidity = amountToUpdate
```

## 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 | contracts/CoinToken.sol#L131,137,143,147,151,155,162,166,173,177,183 |
| 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.

```solidity
function isContract(address account) internal view returns (bool) {
        uint256 size;
        assembly { size := extcodesize(account) }
        return size > 0;
    }

...

        (bool success, ) = recipient.call{ value: amount }("");
        require(success, "Address: unable to send value, recipient may
have reverted");
    }

function functionCall(address target, bytes memory data) internal returns
(bytes memory) {
        return functionCall(target, data, "Address: low-level call failed");
    }

...
```

## 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.

## L13 - Divide before Multiply Operation

| Criticality | Minor / Informative |
|---|---|
| Location | contracts/CoinToken.sol#L497,498 |
| Status | Unresolved |

## Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
_maxTxAmount = (_tTotal * 5 / 1000) * 10 ** _decimals
```

## Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

# L16 - Validate Variable Setters

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/CoinToken.sol#L499,515,516,665 |
| **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.

```
_devWalletAddress = feeaddress
_owner = tokenOwner
payable(service).transfer(msg.value)
_devWalletAddress = _addr
```

## 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.

## L17 - Usage of Solidity Assembly

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/CoinToken.sol#L133,188 |
| **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.

```solidity
assembly { size := extcodesize(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.

# L19 - Stable Compiler Version

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/CoinToken.sol#L11 |
| **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 ^0.8.4;
```

## 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.

# 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 | | | |
| | tryAdd | Internal | | |
| | trySub | Internal | | |
| | tryMul | Internal | | |
| | tryDiv | Internal | | |
| | tryMod | Internal | | |
| | add | Internal | | |
| | sub | Internal | | |
| | mul | Internal | | |
| | div | Internal | | |

| | | | | |
|---|---|---|---|---|
| | mod | Internal | | |
| | sub | Internal | | |
| | div | Internal | | |
| | mod | Internal | | |
| | | | | |
| **Context** | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **Address** | Library | | | |
| | isContract | Internal | | |
| | sendValue | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionStaticCall | Internal | | |
| | functionStaticCall | Internal | | |
| | functionDelegateCall | Internal | ✓ | |
| | functionDelegateCall | Internal | ✓ | |
| | _verifyCallResult | Private | | |
| | | | | |
| **Ownable** | Implementation | Context | | |

| | | Public | ✓ | - |
|---|---|---|---|---|
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | 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 | | - |
| | mint | 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 | ✓ | - |

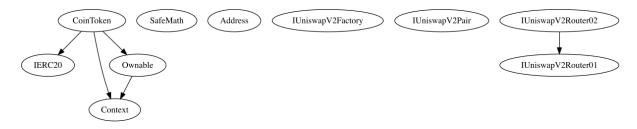| | | | | |
|---|---|---|---|---|
| | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactETHForTokensSupportingFeeOnTransferTokens | External | Payable | - |
| | swapExactTokensForETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | | | | |
| **CoinToken** | Implementation | Context, IERC20, Ownable | | |
| | | Public | Payable | - |
| | 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 | ✓ | |
| excludeFromFee | Public | ✓ | onlyOwner |
| includeInFee | Public | ✓ | onlyOwner |
| setTaxFeePercent | External | ✓ | onlyOwner |
| setDevFeePercent | External | ✓ | onlyOwner |
| setLiquidityFeePercent | External | ✓ | onlyOwner |
| setMaxTxPercent | Public | ✓ | onlyOwner |
| setDevWalletAddress | Public | ✓ | onlyOwner |
| setSwapAndLiquifyEnabled | Public | ✓ | onlyOwner |
| | External | Payable | - |
| _reflectFee | Private | ✓ | |
| _getValues | Private | | |
| _getTValues | Private | | |
| _getRValues | Private | | |
| _getRate | Private | | |
| _getCurrentSupply | Private | | |
| _takeLiquidity | Private | ✓ | |
| _takeDev | Private | ✓ | |
| calculateTaxFee | Private | | |
| calculateDevFee | Private | | |

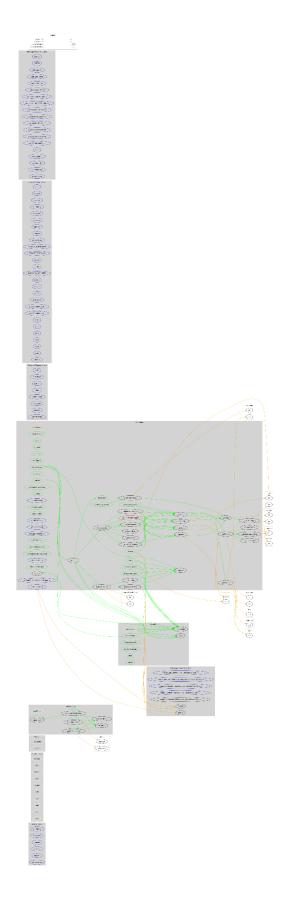| | calculateLiquidityFee | Private | | |
|---|---|---|---|---|
| | removeAllFee | Private | ✓ | |
| | restoreAllFee | Private | ✓ | |
| | isExcludedFromFee | Public | | - |
| | _approve | Private | ✓ | |
| | _transfer | Private | ✓ | |
| | swapAndLiquify | Private | ✓ | lockTheSwap |
| | swapTokensForEth | Private | ✓ | |
| | addLiquidity | Private | ✓ | |
| | _tokenTransfer | Private | ✓ | |
| | _transferStandard | Private | ✓ | |
| | _transferToExcluded | Private | ✓ | |
| | _transferFromExcluded | Private | ✓ | |
| | setRouterAddress | External | ✓ | onlyOwner |
| | setNumTokensSellToAddToLiquidity | External | ✓ | onlyOwner |

# Inheritance Graph

# Flow Graph

# Summary

Chris World Asset 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 stop transactions and 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