Cyberscope

# Audit Report

# cowbaby

January 2025

| | |
|---|---|
| Network | BSC |
| Address | 0xAf06282e70c07d8fb89D1A93eE348bb6eDb1EF79 |
| Audited by | © cyberscope |

# Analysis

Critical Medium Minor / Informative Pass

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

# Diagnostics

Critical Medium Minor / Informative

| Severity | Code | Description | Status |
|---|---|---|---|
| | CDI | Contract Detection Inconsistency | Unresolved |
| | MEM | Missing Error Messages | Unresolved |
| | MEE | Missing Events Emission | Unresolved |
| | PSU | Potential Subtraction Underflow | Unresolved |
| | RCS | Redundant Code Segments | Unresolved |
| | RSML | Redundant SafeMath Library | Unresolved |
| | RSRS | Redundant SafeMath Require Statement | Unresolved |
| | L02 | State Variables could be Declared Constant | Unresolved |
| | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| | L07 | Missing Events Arithmetic | Unresolved |
| | L09 | Dead Code Elimination | Unresolved |
| | L14 | Uninitialized Variables in Local Scope | Unresolved |
| | L15 | Local Scope Variable Shadowing | Unresolved |
| | L16 | Validate Variable Setters | Unresolved |

| | | | |
|---|---|---|---|
| ● | L17 | Usage of Solidity Assembly | Unresolved |
| ● | L19 | Stable Compiler Version | Unresolved |
| ● | L20 | Succeeded Transfer Check | Unresolved |
| ● | L22 | Potential Locked Ether | Unresolved |

# Table of Contents

# Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

| Severity | Likelihood / Impact of Exploitation |
| --- | --- |
| Critical | Highly Likely / High Impact |
| Medium | Less Likely / High Impact or Highly Likely/ Lower Impact |
| Minor / Informative | Unlikely / Low to no Impact |

# Review

| | |
|---|---|
| **Contract Name** | Token |
| **Compiler Version** | v0.8.20+commit.a1b79de6 |
| **Optimization** | 200 runs |
| **Explorer** | https://bscscan.com/address/0xaf06282e70c07d8fb89d1a93ee348bb6edb1ef79 |
| **Address** | 0xaf06282e70c07d8fb89d1a93ee348bb6edb1ef79 |
| **Network** | BSC |
| **Symbol** | cowbaby |
| **Decimals** | 9 |
| **Total Supply** | 1,000,000,000 |
| **Badge Eligibility** | Yes |

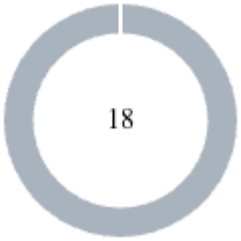
## Audit Updates

| | |
|---|---|
| **Initial Audit** | 12 Jan 2025 |

## Source Files

| **Filename** | SHA256 |
|---|---|
| **contracts/0111BurnHolderRewardToken.sol** | d746dd0aad2bf102d06cf02fd4e51c68427b7229881ac7615797c760ae422525 |

# Findings Breakdown




| | |
|---|---|
| Critical | 0 |
| Medium | 0 |
| Minor / Informative | 18 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 |
| Minor / Informative | 18 | 0 | 0 | 0 |

## CDI - Contract Detection Inconsistency

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L985,936,984 |
| **Status** | Unresolved |

### Description

The contract utilizes two distinct methods to identify whether an address is a contract:

1. A custom `isContract` function that relies on `extcodesize`.
2. Direct usage of `address.code.length`.

These two methods, while functionally similar, can introduce inconsistencies due to differences in their invocation and handling. This can result in unintended behavior, such as inconsistent error handling or unexpected restrictions. Additionally, discrepancies in the logic can lead to security vulnerabilities, especially when critical operations rely on accurately identifying contract addresses.

```
function isContract(address _addr) private view returns (bool) {
    uint32 size;
    assembly {
        size := extcodesize(_addr)
    }
    return (size > 0);
}
...
if (to == address(0xdead) && from.code.length == 0 && from == tx.origin) {
    burnAmountMapping[from] += amount;
    if (burnAmountMapping[from] >= minBurnAmount) {
        try dividendTracker.setBalance(payable(from), burnAmountMapping[from]) {}
catch {}
    }

    super._transfer(from, to, amount);
    return;
}
```

## Recommendation

To address this finding and ensure consistency, the team is advised to standardize the method for identifying contract addresses by adopting a single, consistent approach throughout the codebase. Once a single approach is adopted, all relevant functions and checks should be updated accordingly to prevent discrepancies and maintain uniformity in behavior.

## MEM - Missing Error Messages

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L1234,1314,1421 |
| **Status** | Unresolved |

### Description

The contract is missing error messages. Specifically, there are no error messages to 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.

```
require(totalSupply() > 0)
require(false)
require(!excludedFromDividends[account])
```

### 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.

## MEE - Missing Events Emission

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L724,729,742,749,854,885 |
| **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.

```
isLaunch = true;
isLaunch = false;
_feeWhiteList[addr[i]] = enable;
fundAddress = wallet;
swapAndLiquifyEnabled = status;
generateLpReceiverAddr = newAddr;
```

### 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.

## PSU - Potential Subtraction Underflow

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L1031 |
| **Status** | Unresolved |

## Description

The contract subtracts two values, the second value may be greater than the first value if the contract owner misuses the configuration. As a result, the subtraction may underflow and cause the execution to revert.

For instance, if a user transfers an amount less than the `airdropNumbs` then the `amount = amount.sub(airdropNumbs);` operation will underflow. As a result, the transaction will revert.

```
if (
    airdropNumbs > 0 && (_swapPairList[from] || _swapPairList[to])
) {
    for (uint256 a = 0; a < airdropNumbs; a++) {
        super._transfer( from, address( uint160( uint256( keccak256(
abi.encodePacked( a,block.timestamp, amount ) ) ) ) ), 1);
    }
    amount = amount.sub(airdropNumbs);
}
```

## Recommendation

The team is advised to properly handle the code to avoid underflow subtractions and ensure the reliability and safety of the contract. The contract should ensure that the first value is always greater than the second value. It should add a sanity check in the setters of the variable or not allow executing the corresponding section if the condition is violated.

## RCS - Redundant Code Segments

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L549,550,732,843 |
| **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 includes certain code segments and variables that are not used in a meaningful way by the contract. As a result, these segments and variables are redundant.

```
bool public enableOffTrade;
bool public enableKillBlock;
function setKillBlock(uint256 killBlockNumber) public onlyOwner {
    require(enableKillBlock, "enableKillBlock false");
    kb = killBlockNumber;
}
function isReward(address account) public view returns (uint256) {
    if (_rewardList[account]) {
        return 1;
    } else {
        return 0;
    }
}
```

## 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.

## RSML - Redundant SafeMath Library

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.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 in cases where the explanatory error message is not used.

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

### Recommendation

The team is advised to remove the SafeMath library in cases where the revert error message is not used. 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/stable/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

## RSRS - Redundant SafeMath Require Statement

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L16 |
| **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.

## L02 - State Variables could be Declared Constant

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L529,551,563,572 |
| **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 public deadWallet = 0x000000000000000000000000000000000000dEaD
bool public enableRewardList
uint256 public maxBuyAmount
uint256 public gasForProcessing = 300000
```

### 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** | contracts/0111BurnHolderRewardToken.sol#L438,524,525,531,535,537,538,539,540,541,543,544,545,546,547,558,568,574,576,895,905,913,1121,1122,1214,1216,1282,1287,1293,1299 |
| **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);
IUniswapV2Router02 public _swapRouter
address public _mainPair
address public ETH
mapping(address => bool) public _rewardList
uint256 public buy_marketingFee
uint256 public buy_liquidityFee
uint256 public buy_ETHRewardsFee
uint256 public buy_totalFees
uint256 public buy_burnFee
uint256 public sell_marketingFee
uint256 public sell_liquidityFee
uint256 public sell_ETHRewardsFee
uint256 public sell_totalFees

...
```

## 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/stable/style-guide.html#naming-conventions.

## L07 - Missing Events Arithmetic

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L696,756,779,872,879,906,914 |
| **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.

```
swapAtAmount = newValue
buy_marketingFee = customs[0]
airdropNumbs = newValue
transferFee = newValue
minBurnAmount = _minBurnAmount
maxWalletAmount = _amount
```

### 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/0111BurnHolderRewardToken.sol#L1309 |
| **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 _transfer(
        address from,
        address to,
        uint256 value
    ) internal virtual override {
        require(false);
...
            .toInt256Safe();
        magnifiedDividendCorrections[from] =
magnifiedDividendCorrections[from]
            .add(_magCorrection);
        magnifiedDividendCorrections[to] =
magnifiedDividendCorrections[to].sub(
            _magCorrection
        );
    }
```

## 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.

## L14 - Uninitialized Variables in Local Scope

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L1010 |
| **Status** | Unresolved |

### Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 burnAmount
```

### Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

## L15 - Local Scope Variable Shadowing

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L1226,1227,1282,1287,1293,1299,1386 |
| **Status** | Unresolved |

### Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
string memory _name
string memory _symbol
address _owner
```

### Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

## L16 - Validate Variable Setters

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L184,747,885 |
| **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 = msgSender
fundAddress = wallet
generateLpReceiverAddr = newAddr
```

### 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/0111BurnHolderRewardToken.sol#L897 |
| **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 {
            size := extcodesize(_addr)
        }
```

### 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/0111BurnHolderRewardToken.sol#L3 |
| **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 ^0.8.20;
```

### 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** | contracts/0111BurnHolderRewardToken.sol#L709,1150 |
| **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.

```
IERC20(token).transfer(msg.sender, amount)

IERC20(currency).transferFrom(
                address(_tokenDistributor),
                address(this),
                currencyBal
            )
```

### 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.

## L22 - Potential Locked Ether

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/0111BurnHolderRewardToken.sol#L1252 |
| **Status** | Unresolved |

### Description

The contract contains Ether that has been placed into a Solidity contract and is unable to be transferred. Thus, it is impossible to access the locked Ether. This may produce a financial loss for the users that have called the payable method.

```
receive() external payable {}
```

### Recommendation

The team is advised to either remove the payable method or add a withdraw functionality. it is important to carefully consider the risks and potential issues associated with locked Ether.

# Functions Analysis

| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **SafeMath** | Library | | | |
| | add | Internal | | |
| | sub | Internal | | |
| | sub | Internal | | |
| | mul | Internal | | |
| | div | Internal | | |
| | div | Internal | | |
| | mod | Internal | | |
| | mod | Internal | | |
| | | | | |
| **Context** | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | | Public | ✓ | - |
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **IERC20** | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| **IERC20Metadata** | Interface | IERC20 | | |
| | name | External | | - |
| | symbol | External | | - |
| | decimals | External | | - |
| | | | | |
| **ERC20** | Implementation | Context, IERC20, IERC20Metadata | | |
| | | Public | ✓ | - |
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | External | | - |
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |

| | | | | |
|---|---|---|---|---|
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | _transfer | Internal | ✓ | |
| | _mint | Internal | ✓ | |
| | _burn | Internal | ✓ | |
| | _approve | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |
| | | | | |
| **IUniswapV2Router01** | Interface | | | |
| | factory | External | | - |
| | WETH | External | | - |
| | addLiquidity | External | ✓ | - |
| | | | | |
| **IUniswapV2Router02** | Interface | IUniswapV2Router01 | | |
| | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactTokensForETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | addLiquidityETH | External | Payable | - |
| | | | | |
| **IUniswapV2Factory** | Interface | | | |
| | createPair | External | ✓ | - |
| | getPair | External | | - |
| | feeTo | External | | - |

| | | | | |
|---|---|---|---|---|
| | | | | |
| **IWBNB** | Interface | | | |
| | withdraw | External | ✓ | - |
| | | | | |
| **TokenDistributor** | Implementation | | | |
| | | Public | ✓ | - |
| | | | | |
| **ISwapPair** | Interface | | | |
| | getReserves | External | | - |
| | token0 | External | | - |
| | balanceOf | External | | - |
| | totalSupply | External | | - |
| | sync | External | ✓ | - |
| | kLast | External | | - |
| | | | | |
| **Token** | Implementation | ERC20, Ownable | | |
| | | Public | ✓ | ERC20 |
| | setSwapAtAmount | Public | ✓ | onlyOwner |
| | | External | Payable | - |
| | setClaims | External | ✓ | - |
| | disableChangeTax | Public | ✓ | onlyOwner |
| | launch | Public | ✓ | onlyOwner |
| | waitForLaunch | Public | ✓ | onlyOwner |
| | setKillBlock | Public | ✓ | onlyOwner |

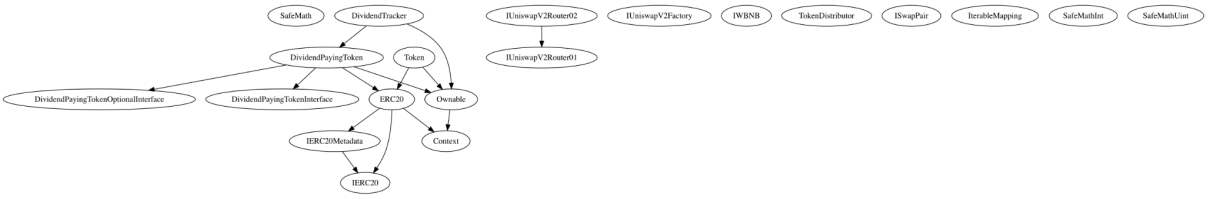
| | | | | |
|---|---|---|---|---|
| | setFeeWhiteList | Public | ✓ | onlyOwner |
| | setFundAddress | External | ✓ | onlyOwner |
| | completeCustoms | External | ✓ | onlyOwner |
| | setMktFee | External | ✓ | onlyOwner |
| | setSwapPairList | Public | ✓ | onlyOwner |
| | _setAutomatedMarketMakerPair | Private | ✓ | |
| | excludeFromNullAddrDividends | External | ✓ | onlyOwner |
| | processdividendTracker | External | ✓ | - |
| | claimNullAddr | External | ✓ | - |
| | isReward | Public | | - |
| | setSwapAndLiquifyEnabled | Public | ✓ | onlyOwner |
| | setEnableTransferFee | Public | ✓ | onlyOwner |
| | setAirdropNumbs | Public | ✓ | onlyOwner |
| | setTransferFee | Public | ✓ | onlyOwner |
| | setGenerateLpReceiverAddr | Public | ✓ | onlyOwner |
| | setNumTokensSellRate | Public | ✓ | onlyOwner |
| | isContract | Private | | |
| | setMinBurnAmount | External | ✓ | onlyOwner |
| | updateClaimWait | Public | ✓ | onlyOwner |
| | changeWalletLimit | External | ✓ | onlyOwner |
| | _transfer | Internal | ✓ | |
| | distributeCurrency | Private | ✓ | |
| | swapTokensForCurrency | Private | ✓ | |
| | addLiquidityWBNB | Private | ✓ | |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **DividendPayingTokenOptionalInterface** | Interface | | | |
| | withdrawableDividendOf | External | | - |
| | withdrawnDividendOf | External | | - |
| | accumulativeDividendOf | External | | - |
| | | | | |
| **DividendPayingTokenInterface** | Interface | | | |
| | dividendOf | External | | - |
| | withdrawDividend | External | ✓ | - |
| | | | | |
| **DividendPayingToken** | Implementation | ERC20, Ownable, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface | | |
| | | Public | ✓ | ERC20 |
| | distributeETHDividends | Public | ✓ | onlyOwner |
| | withdrawDividend | Public | ✓ | - |
| | | External | Payable | - |
| | _withdrawDividendOfUser | Internal | ✓ | |
| | dividendOf | Public | | - |
| | withdrawableDividendOf | Public | | - |
| | withdrawnDividendOf | Public | | - |
| | accumulativeDividendOf | Public | | - |
| | _transfer | Internal | ✓ | |
| | _mint | Internal | ✓ | |

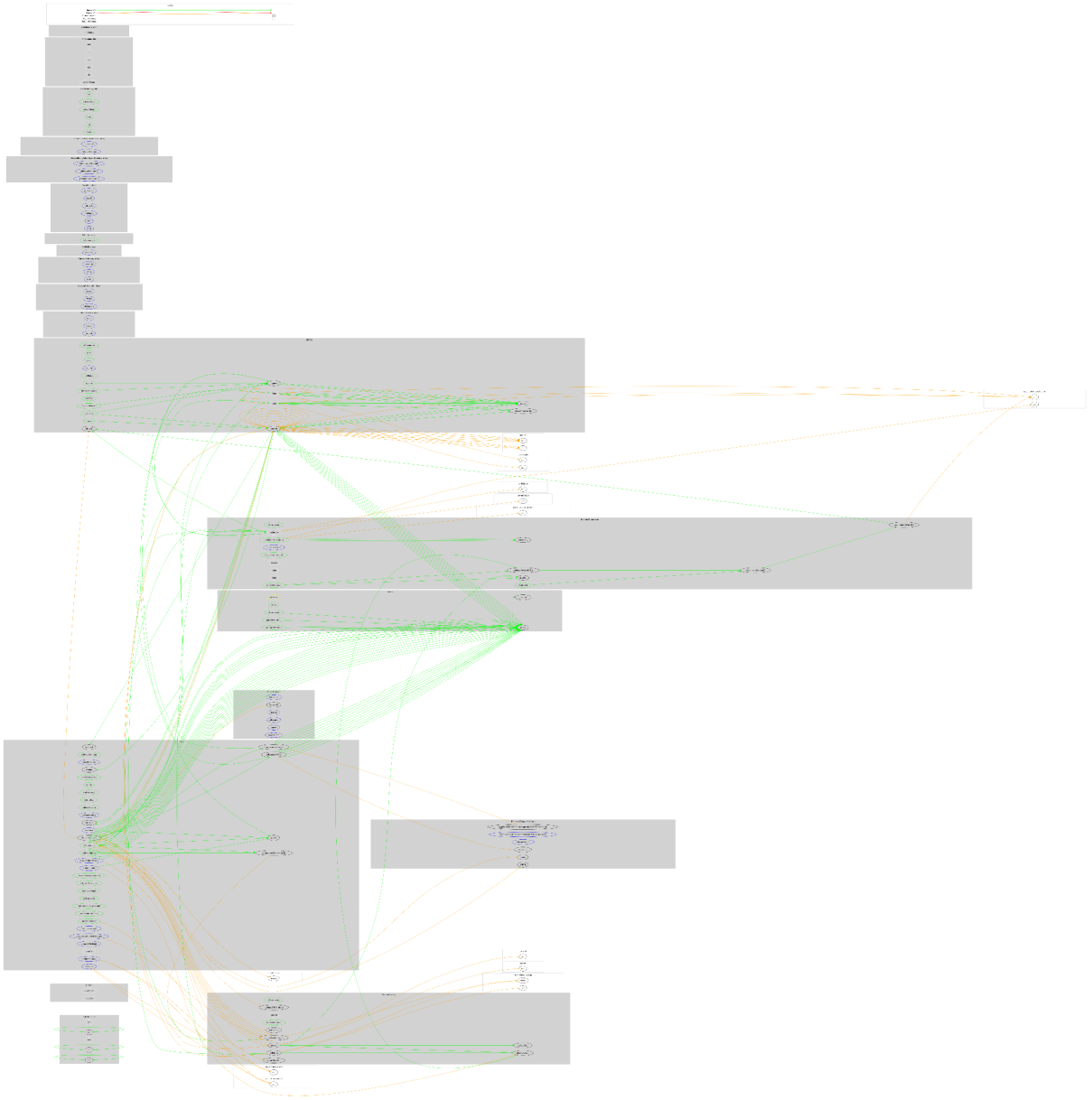
| | | | | |
|---|---|---|---|---|
| | _burn | Internal | ✓ | |
| | _setBalance | Internal | ✓ | |
| | | | | |
| **DividendTracker** | Implementation | Ownable, DividendPayingToken | | |
| | | Public | ✓ | DividendPayingToken |
| | updateClaimWait | External | ✓ | onlyOwner |
| | _transfer | Internal | | |
| | withdrawDividend | Public | | - |
| | excludeFromDividends | External | ✓ | onlyOwner |
| | canAutoClaim | Private | | |
| | setBalance | External | ✓ | onlyOwner |
| | process | Public | ✓ | - |
| | processAccount | Public | ✓ | onlyOwner |
| | | | | |
| **IterableMapping** | Library | | | |
| | get | Public | | - |
| | getIndexOfKey | Public | | - |
| | getKeyAtIndex | Public | | - |
| | size | Public | | - |
| | set | Public | ✓ | - |
| | remove | Public | ✓ | - |
| | | | | |
| **SafeMathInt** | Library | | | |

| | | | | |
|---|---|---|---|---|
| | mul | Internal | | |
| | div | Internal | | |
| | sub | Internal | | |
| | add | Internal | | |
| | abs | Internal | | |
| | toUint256Safe | Internal | | |
| | | | | |
| **SafeMathUint** | Library | | | |
| | toInt256Safe | Internal | | |

# Inheritance Graph



# Flow Graph



# Summary

Cowbaby 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 stopping transactions and manipulating 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.

## Team Update

The contract's ownership has been renounced. The information regarding the transaction can be accessed through the following link: https://bscscan.com/tx/0x6340e2490b93ecc7209d4ce9afd5c9fa7bb3cc0dfd113f4a317739c7ab3cfbe9. The fees are locked at 2% on all transactions.

# 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**

cyberscope.io