BLOCKHAT
SECURITY

Rewardo Token

Smart Contract Security Audit

Prepared by BlockHat

April 9th, 2024 – April 14th, 2024

BlockHat.io

contact@blockhat.io

Document Properties

Client	Rewardo Token
Version	0.1
Classification	Private

Scope

The Rewardo Token Contract in the Rewardo Token Repository

Link	Address
https://etherscan.io/address/0x797091E1f6c9Ce7BEb7dd6Ff1e4e4fBDe8fc0A06	0x797091E1f6c9Ce7BEb7dd6Ff1e4e4fBDe8fc0A06

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

Post-audit notes by Rewardo Team in the end of this report.

1 Introduction

Rewardo Token engaged BlockHat to conduct a security assessment on the Rewardo Token beginning on April 9th, 2024 and ending April 14th, 2024. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About Rewardo Token

Issuer	Rewardo Token
Website	`https://www.rewardotoken.com`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

BlockHat used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by BlockHat are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low
		Likelihood		

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the Rewardo Token implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 1 high-severity, 2 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Non-Withdrawable Ether Generated swapAndLiquify Function	HIGH	Not fixed
Use of Outdated ERC20 Implementation	LOW	Not Fixed
Unnecessary Override in _beforeTokenTransfer Function	LOW	Not Fixed

Note: Please find post audit explanations at the end of this audit.

3 Finding Details

A token.sol

A.1 Non-Withdrawable Ether Generated swapAndLiquify Function [HIGH]

Description:

Through the swapAndLiquify function, the contract acquires non-withdrawable ether by converting half of its contractTokenBalance tokens to ether. The remaining half of the tokens, along with a portion of the converted ether, are deposited into the -ether pool as liquidity during the swap. With each call of the swapAndLiquify function, a small amount of ether is left in the contract because the token price decreases after swapping the first half of tokens for ether. Additionally, the remaining half of tokens requires less converted ether to be paired with it during liquidity addition. As a result, the contract does not seem to offer any way to withdraw the acquired ether, which will remain locked within the contract permanently.

Risk Level:

Likelihood – 3
Impact - 5

Recommendation:

We suggest adding a withdraw function within the contract to enable ether withdrawals. Another option could be to distribute the ether proportionally to the token holders based on the number of tokens they hold. Alternatively, the leftover ether could be used to purchase tokens from the market to increase their price.

Status - Not fixed

A.2 Use of Outdated ERC20 Implementation [LOW]

Description:

The contract implements its own version of the ERC20 standard, which may not include the latest security practices, optimizations, and features found in widely-used libraries such as OpenZeppelin's ERC20 implementation. Using an outdated or custom implementation can introduce risks and compatibility issues with other contracts and decentralized applications (dApps).

Recommendation:

Replace the custom ERC20 implementation with the latest version of the ERC20 token standard from a reputable library such as OpenZeppelin. This not only ensures compliance with the latest security practices but also benefits from the community's scrutiny, ongoing maintenance, and updates.

Status - Not Fixed

A.3 Unnecessary Override in _beforeTokenTransfer Function [LOW]

Description:

The function _beforeTokenTransfer overrides an inherited function but does not implement any additional logic beyond the parent class's implementation. The function simply calls super._beforeTokenTransfer(from, to, amount), directly invoking the inherited method without modification. This redundant override could lead to confusion and increased maintenance overhead without providing any functional benefit.

Code:

Listing 1: token.sol

```
    function _beforeTokenTransfer(address from, address to, uint256
        ↪ amount)
        internal
        override
    {
        super._beforeTokenTransfer(from, to, amount);
    }
```

Risk Level:

Likelihood – 1
Impact - 1

Recommendation:

It is advisable to remove the override of the _beforeTokenTransfer function if no additional logic is required beyond what is implemented in the parent class

Status - Not Fixed

4 Static Analysis (Slither)

Description:

Block Hat expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
INFO:Detectors:
Rewardo_Token._addLiquidity(uint256,uint256) (Token.sol#207-211) sends
    ↪ eth to arbitrary user
        Dangerous calls:
        - routerV2.addLiquidityETH{value: coinAmount}(address(this),
            ↪ tokenAmount,0,0,address(0),block.timestamp) (Token.sol
            ↪ #210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in Rewardo_Token._transfer(address,address,uint256) (Token.
    ↪ sol#295-374):
        External calls:
        - _swapTokensForOtherTokens(token2Swap) (Token.sol#340)
                - routerV2.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (tokenAmount,0,path,address(this),block.timestamp)
                    ↪ (Token.sol#138)
        - success = _sendInOtherTokens(marketingAddress,marketingPortion)
            ↪  (Token.sol#345)
                - feeToken.transfer(to,amount) (Token.sol#127)
```

```
           - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                  - routerV2.addLiquidityETH{value: coinAmount}(address(this
                     ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                     ↪ Token.sol#210)
                  - routerV2.
                     ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                     ↪ tokenAmount,0,path,address(this),block.timestamp) (
                     ↪ Token.sol#184)
           External calls sending eth:
           - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                  - routerV2.addLiquidityETH{value: coinAmount}(address(this
                     ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                     ↪ Token.sol#210)
           State variables written after the call(s):
           - _liquidityPending = 0 (Token.sol#356)
           Rewardo_Token._liquidityPending (Token.sol#27) can be used in
               ↪ cross function reentrancies:
           - Rewardo_Token._transfer(address,address,uint256) (Token.sol
               ↪ #295-374)
           - Rewardo_Token.getAllPending() (Token.sol#152-154)
    Reentrancy in Rewardo_Token._transfer(address,address,uint256) (Token.
        ↪ sol#295-374):
           External calls:
           - _swapTokensForOtherTokens(token2Swap) (Token.sol#340)
                  - routerV2.
                     ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                     ↪ (tokenAmount,0,path,address(this),block.timestamp)
                     ↪ (Token.sol#138)
           - success = _sendInOtherTokens(marketingAddress,marketingPortion)
               ↪  (Token.sol#345)
                  - feeToken.transfer(to,amount) (Token.sol#127)
           - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                  - routerV2.addLiquidityETH{value: coinAmount}(address(this
                     ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                ↪ Token.sol#210)
            - routerV2.
                ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                ↪ tokenAmount,0,path,address(this),block.timestamp) (
                ↪ Token.sol#184)
    - _sendDividends(_rewardsPending) (Token.sol#360)
            - success = IERC20(rewardToken).approve(address(
                ↪ dividendTracker),dividends) (Token.sol#249)
            - routerV2.
                ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                ↪ (tokenAmount,0,path,address(this),block.timestamp)
                ↪ (Token.sol#240)
            - dividendTracker.distributeDividends(dividends) (Token.
                ↪ sol#252)
    External calls sending eth:
    - _swapAndLiquify(_liquidityPending) (Token.sol#355)
            - routerV2.addLiquidityETH{value: coinAmount}(address(this
                ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                ↪ Token.sol#210)
    State variables written after the call(s):
    - super._transfer(from,to,amount) (Token.sol#367)
            - _balances[from] = fromBalance - amount (ERC20.sol#231)
            - _balances[to] += amount (ERC20.sol#234)
    ERC20._balances (ERC20.sol#39) can be used in cross function
        ↪ reentrancies:
    - ERC20._burn(address,uint256) (ERC20.sol#277-293)
    - ERC20._mint(address,uint256) (ERC20.sol#251-264)
    - ERC20._transfer(address,address,uint256) (ERC20.sol#222-240)
    - ERC20.balanceOf(address) (ERC20.sol#101-103)
    - _rewardsPending = 0 (Token.sol#361)
    Rewardo_Token._rewardsPending (Token.sol#28) can be used in cross
        ↪  function reentrancies:
    - Rewardo_Token._transfer(address,address,uint256) (Token.sol
        ↪ #295-374)
```

```
          - Rewardo_Token.getAllPending() (Token.sol#152-154)
          - _swapping = false (Token.sol#364)
          Rewardo_Token._swapping (Token.sol#40) can be used in cross
             ↪ function reentrancies:
          - Rewardo_Token._transfer(address,address,uint256) (Token.sol
             ↪ #295-374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #reentrancy-vulnerabilities
INFO:Detectors:
DividendPayingToken.distributeDividends(uint256) (TokenDividendTracker.
   ↪ sol#188-202) ignores return value by IERC20(rewardToken).
   ↪ transferFrom(msg.sender,address(this),amount) (
   ↪ TokenDividendTracker.sol#192)
DividendPayingToken._withdrawDividend(address) (TokenDividendTracker.sol
   ↪ #204-223) ignores return value by IERC20(rewardToken).transfer(
   ↪ account,withdrawableDividend) (TokenDividendTracker.sol#210-219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #unchecked-transfer
INFO:Detectors:
Rewardo_Token._transfer(address,address,uint256) (Token.sol#295-374)
   ↪ uses a Boolean constant improperly:
          -false || _marketingPending > 0 (Token.sol#336)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #misuse-of-a-boolean-constant
INFO:Detectors:
Rewardo_Token._transfer(address,address,uint256) (Token.sol#295-374)
   ↪ performs a multiplication on the result of a division:
          - fees = amount * totalFees[txType] / 10000 (Token.sol#314)
          - _marketingPending += fees * marketingFees[txType] / totalFees[
             ↪ txType] (Token.sol#317)
Rewardo_Token._transfer(address,address,uint256) (Token.sol#295-374)
   ↪ performs a multiplication on the result of a division:
          - fees = amount * totalFees[txType] / 10000 (Token.sol#314)
```

```
        - _liquidityPending += fees * liquidityFees[txType] / totalFees[
            ↪ txType] (Token.sol#319)
Rewardo_Token._transfer(address,address,uint256) (Token.sol#295-374)
    ↪ performs a multiplication on the result of a division:
        - fees = amount * totalFees[txType] / 10000 (Token.sol#314)
        - _rewardsPending += fees * rewardsFees[txType] / totalFees[
            ↪ txType] (Token.sol#321)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #divide-before-multiply
INFO:Detectors:
Reentrancy in Rewardo_Token._transfer(address,address,uint256) (Token.
    ↪ sol#295-374):
        External calls:
        - _swapTokensForOtherTokens(token2Swap) (Token.sol#340)
                - routerV2.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (tokenAmount,0,path,address(this),block.timestamp)
                    ↪ (Token.sol#138)
        - success = _sendInOtherTokens(marketingAddress,marketingPortion)
            ↪  (Token.sol#345)
                - feeToken.transfer(to,amount) (Token.sol#127)
        State variables written after the call(s):
        - _marketingPending = 0 (Token.sol#350)
        Rewardo_Token._marketingPending (Token.sol#26) can be used in
            ↪ cross function reentrancies:
        - Rewardo_Token._transfer(address,address,uint256) (Token.sol
            ↪ #295-374)
        - Rewardo_Token.getAllPending() (Token.sol#152-154)
Reentrancy in DividendPayingToken._withdrawDividend(address) (
    ↪ TokenDividendTracker.sol#204-223):
        External calls:
        - IERC20(rewardToken).transfer(account,withdrawableDividend) (
            ↪ TokenDividendTracker.sol#210-219)
        State variables written after the call(s):
```

```
        - withdrawnDividends[account] = withdrawnDividends[account] -
           ↪ withdrawableDividend (TokenDividendTracker.sol#215)
        DividendPayingToken.withdrawnDividends (TokenDividendTracker.sol
           ↪ #180) can be used in cross function reentrancies:
        - DividendPayingToken.withdrawableDividendOf(address) (
           ↪ TokenDividendTracker.sol#229-231)
        - DividendPayingToken.withdrawnDividendOf(address) (
           ↪ TokenDividendTracker.sol#233-235)
        - withdrawnDividends[account] = withdrawnDividends[account] -
           ↪ withdrawableDividend (TokenDividendTracker.sol#218)
        DividendPayingToken.withdrawnDividends (TokenDividendTracker.sol
           ↪ #180) can be used in cross function reentrancies:
        - DividendPayingToken.withdrawableDividendOf(address) (
           ↪ TokenDividendTracker.sol#229-231)
        - DividendPayingToken.withdrawnDividendOf(address) (
           ↪ TokenDividendTracker.sol#233-235)
   Reentrancy in DividendTracker.process(uint256) (TokenDividendTracker.sol
      ↪ #455-492):
        External calls:
        - claim(account) (TokenDividendTracker.sol#475)
              - IERC20(rewardToken).transfer(account,
                 ↪ withdrawableDividend) (TokenDividendTracker.sol
                 ↪ #210-219)
        State variables written after the call(s):
        - lastProcessedIndex = _lastProcessedIndex (TokenDividendTracker.
           ↪ sol#489)
        DividendTracker.lastProcessedIndex (TokenDividendTracker.sol#324)
           ↪  can be used in cross function reentrancies:
        - DividendTracker.getAccountData(address) (TokenDividendTracker.
           ↪ sol#376-405)
        - DividendTracker.lastProcessedIndex (TokenDividendTracker.sol
           ↪ #324)
        - DividendTracker.process(uint256) (TokenDividendTracker.sol
           ↪ #455-492)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-1
INFO:Detectors:
DividendPayingToken._withdrawDividend(address).result (
    ↪ TokenDividendTracker.sol#210) is a local variable never
    ↪ initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-local-variables
INFO:Detectors:
Rewardo_Token._transfer(address,address,uint256) (Token.sol#295-374)
    ↪ ignores return value by dividendTracker.process(gasForProcessing)
    ↪  (Token.sol#372)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return
INFO:Detectors:
DividendPayingToken.constructor(string,string)._name (
    ↪ TokenDividendTracker.sol#186) shadows:
        - TruncatedERC20._name (TokenDividendTracker.sol#28) (state
           ↪ variable)
DividendPayingToken.constructor(string,string)._symbol (
    ↪ TokenDividendTracker.sol#186) shadows:
        - TruncatedERC20._symbol (TokenDividendTracker.sol#29) (state
           ↪ variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #local-variable-shadowing
INFO:Detectors:
Ownable2Step.transferOwnership(address).newOwner (Ownable2Step.sol#35)
    ↪ lacks a zero-check on :
               - _pendingOwner = newOwner (Ownable2Step.sol#36)
DividendTracker.setRewardToken(address)._rewardToken (
    ↪ TokenDividendTracker.sol#341) lacks a zero-check on :
               - rewardToken = _rewardToken (TokenDividendTracker.sol
                  ↪ #344)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation
INFO:Detectors:
DividendPayingToken._withdrawDividend(address) (TokenDividendTracker.sol
    ↪ #204-223) has external calls inside a loop: IERC20(rewardToken).
    ↪ transfer(account,withdrawableDividend) (TokenDividendTracker.sol
    ↪ #210-219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ /#calls-inside-a-loop
INFO:Detectors:
Reentrancy in Rewardo_Token._setAMMPair(address,bool) (Token.sol
    ↪ #395-406):
        External calls:
        - _excludeFromDividends(pair,true) (Token.sol#399)
                - dividendTracker.excludeFromDividends(account,balanceOf(
                    ↪ account),isExcluded) (Token.sol#263)
        State variables written after the call(s):
        - _excludeFromLimits(pair,true) (Token.sol#401)
                - isExcludedFromLimits[account] = isExcluded (Token.sol
                    ↪ #413)
Reentrancy in DividendTrackerFunctions._setRewardToken(address) (
    ↪ TokenDividendTracker.sol#511-515):
        External calls:
        - dividendTracker.setRewardToken(_rewardToken) (
            ↪ TokenDividendTracker.sol#512)
        State variables written after the call(s):
        - rewardToken = _rewardToken (TokenDividendTracker.sol#514)
Reentrancy in Rewardo_Token._swapAndLiquify(uint256) (Token.sol#187-205)
    ↪ :
        External calls:
        - _swapTokensForCoin(halfAmount) (Token.sol#192)
                - routerV2.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ Token.sol#184)
            - (amountToken,amountCoin,liquidity) = _addLiquidity(otherHalf,
               ↪ coinBalance) (Token.sol#197)
                    - routerV2.addLiquidityETH{value: coinAmount}(address(this
                       ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                       ↪ Token.sol#210)
            External calls sending eth:
            - (amountToken,amountCoin,liquidity) = _addLiquidity(otherHalf,
               ↪ coinBalance) (Token.sol#197)
                    - routerV2.addLiquidityETH{value: coinAmount}(address(this
                       ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                       ↪ Token.sol#210)
            State variables written after the call(s):
            - (amountToken,amountCoin,liquidity) = _addLiquidity(otherHalf,
               ↪ coinBalance) (Token.sol#197)
                    - _allowances[owner][spender] = amount (ERC20.sol#312)
    Reentrancy in Rewardo_Token._transfer(address,address,uint256) (Token.
       ↪ sol#295-374):
            External calls:
            - _swapTokensForOtherTokens(token2Swap) (Token.sol#340)
                    - routerV2.
                       ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                       ↪ (tokenAmount,0,path,address(this),block.timestamp)
                       ↪ (Token.sol#138)
            - success = _sendInOtherTokens(marketingAddress,marketingPortion)
               ↪  (Token.sol#345)
                    - feeToken.transfer(to,amount) (Token.sol#127)
            - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                    - routerV2.addLiquidityETH{value: coinAmount}(address(this
                       ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                       ↪ Token.sol#210)
                    - routerV2.
                       ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                       ↪ tokenAmount,0,path,address(this),block.timestamp) (
                       ↪ Token.sol#184)
            External calls sending eth:
            - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                    - routerV2.addLiquidityETH{value: coinAmount}(address(this
                       ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                       ↪ Token.sol#210)
            State variables written after the call(s):
            - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                    - _allowances[owner][spender] = amount (ERC20.sol#312)
    Reentrancy in Rewardo_Token._transfer(address,address,uint256) (Token.
       ↪ sol#295-374):
            External calls:
            - _swapTokensForOtherTokens(token2Swap) (Token.sol#340)
                    - routerV2.
                       ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                       ↪ (tokenAmount,0,path,address(this),block.timestamp)
                       ↪ (Token.sol#138)
            - success = _sendInOtherTokens(marketingAddress,marketingPortion)
               ↪  (Token.sol#345)
                    - feeToken.transfer(to,amount) (Token.sol#127)
            - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                    - routerV2.addLiquidityETH{value: coinAmount}(address(this
                       ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                       ↪ Token.sol#210)
                    - routerV2.
                       ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                       ↪ tokenAmount,0,path,address(this),block.timestamp) (
                       ↪ Token.sol#184)
            - _sendDividends(_rewardsPending) (Token.sol#360)
                    - success = IERC20(rewardToken).approve(address(
                       ↪ dividendTracker),dividends) (Token.sol#249)
                    - routerV2.
                       ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                       ↪ (tokenAmount,0,path,address(this),block.timestamp)
                    ↪ (Token.sol#240)
                - dividendTracker.distributeDividends(dividends) (Token.
                    ↪ sol#252)
            External calls sending eth:
            - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this
                    ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                    ↪ Token.sol#210)
            State variables written after the call(s):
            - _sendDividends(_rewardsPending) (Token.sol#360)
                - _allowances[owner][spender] = amount (ERC20.sol#312)
    Reentrancy in Rewardo_Token._updateRouterV2(address) (Token.sol#376-387)
        ↪ :
            External calls:
            - pairV2 = IUniswapV2Factory(routerV2.factory()).createPair(
                ↪ address(this),routerV2.WETH()) (Token.sol#378)
            - _excludeFromDividends(router,true) (Token.sol#380)
                - dividendTracker.excludeFromDividends(account,balanceOf(
                    ↪ account),isExcluded) (Token.sol#263)
            State variables written after the call(s):
            - _excludeFromLimits(router,true) (Token.sol#382)
                - isExcludedFromLimits[account] = isExcluded (Token.sol
                    ↪ #413)
    Reentrancy in Rewardo_Token._updateRouterV2(address) (Token.sol#376-387)
        ↪ :
            External calls:
            - pairV2 = IUniswapV2Factory(routerV2.factory()).createPair(
                ↪ address(this),routerV2.WETH()) (Token.sol#378)
            - _excludeFromDividends(router,true) (Token.sol#380)
                - dividendTracker.excludeFromDividends(account,balanceOf(
                    ↪ account),isExcluded) (Token.sol#263)
            - _setAMMPair(pairV2,true) (Token.sol#384)
                - dividendTracker.excludeFromDividends(account,balanceOf(
                    ↪ account),isExcluded) (Token.sol#263)
```

```
          State variables written after the call(s):
          - _setAMMPair(pairV2,true) (Token.sol#384)
                 - AMMPairs[pair] = isPair (Token.sol#396)
          - _setAMMPair(pairV2,true) (Token.sol#384)
                 - isExcludedFromLimits[account] = isExcluded (Token.sol
                    ↪ #413)
   Reentrancy in DividendTracker.claim(address) (TokenDividendTracker.sol
      ↪ #425-433):
          External calls:
          - amount = _withdrawDividend(account) (TokenDividendTracker.sol
             ↪ #426)
                 - IERC20(rewardToken).transfer(account,
                    ↪ withdrawableDividend) (TokenDividendTracker.sol
                    ↪ #210-219)
          State variables written after the call(s):
          - lastClaimTimes[account] = block.timestamp (TokenDividendTracker
             ↪ .sol#429)
   Reentrancy in DividendPayingToken.distributeDividends(uint256) (
      ↪ TokenDividendTracker.sol#188-202):
          External calls:
          - IERC20(rewardToken).transferFrom(msg.sender,address(this),
             ↪ amount) (TokenDividendTracker.sol#192)
          State variables written after the call(s):
          - magnifiedDividendPerShare = magnifiedDividendPerShare + (
             ↪ received * magnitude / totalSupply()) (
             ↪ TokenDividendTracker.sol#196)
          - totalDividendsDistributed = totalDividendsDistributed +
             ↪ received (TokenDividendTracker.sol#200)
   Reentrancy in Rewardo_Token.initialize(address,address,address) (Token.
      ↪ sol#109-114):
          External calls:
          - _setRewardToken(_rewardToken) (Token.sol#111)
                 - dividendTracker.setRewardToken(_rewardToken) (
                    ↪ TokenDividendTracker.sol#512)
          - _updateRouterV2(_router) (Token.sol#113)
                 - dividendTracker.excludeFromDividends(account,balanceOf(
                    ↪ account),isExcluded) (Token.sol#263)
                 - pairV2 = IUniswapV2Factory(routerV2.factory()).
                    ↪ createPair(address(this),routerV2.WETH()) (Token.
                    ↪ sol#378)
          State variables written after the call(s):
          - _updateRouterV2(_router) (Token.sol#113)
                 - AMMPairs[pair] = isPair (Token.sol#396)
          - _updateRouterV2(_router) (Token.sol#113)
                 - isExcludedFromLimits[account] = isExcluded (Token.sol
                    ↪ #413)
   Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
      ↪ #reentrancy-vulnerabilities-2
   INFO:Detectors:
   Reentrancy in Rewardo_Token._sendDividends(uint256) (Token.sol#243-256):
          External calls:
          - _swapTokensForOtherRewardTokens(tokenAmount) (Token.sol#244)
                 - routerV2.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (tokenAmount,0,path,address(this),block.timestamp)
                    ↪ (Token.sol#240)
          - success = IERC20(rewardToken).approve(address(dividendTracker),
             ↪ dividends) (Token.sol#249)
          - dividendTracker.distributeDividends(dividends) (Token.sol#252)
          Event emitted after the call(s):
          - rewardsFeeSent(dividends) (Token.sol#253)
   Reentrancy in Rewardo_Token._setAMMPair(address,bool) (Token.sol
      ↪ #395-406):
          External calls:
          - _excludeFromDividends(pair,true) (Token.sol#399)
                 - dividendTracker.excludeFromDividends(account,balanceOf(
                    ↪ account),isExcluded) (Token.sol#263)
          Event emitted after the call(s):
```

```
          - AMMPairsUpdated(pair,isPair) (Token.sol#405)
          - ExcludeFromLimits(account,isExcluded) (Token.sol#415)
                - _excludeFromLimits(pair,true) (Token.sol#401)
    Reentrancy in Rewardo_Token._swapAndLiquify(uint256) (Token.sol#187-205)
       ↪ :
          External calls:
          - _swapTokensForCoin(halfAmount) (Token.sol#192)
                - routerV2.
                   ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                   ↪ tokenAmount,0,path,address(this),block.timestamp) (
                   ↪ Token.sol#184)
          - (amountToken,amountCoin,liquidity) = _addLiquidity(otherHalf,
             ↪ coinBalance) (Token.sol#197)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this
                   ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                   ↪ Token.sol#210)
          External calls sending eth:
          - (amountToken,amountCoin,liquidity) = _addLiquidity(otherHalf,
             ↪ coinBalance) (Token.sol#197)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this
                   ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                   ↪ Token.sol#210)
          Event emitted after the call(s):
          - Approval(owner,spender,amount) (ERC20.sol#313)
                - (amountToken,amountCoin,liquidity) = _addLiquidity(
                   ↪ otherHalf,coinBalance) (Token.sol#197)
          - liquidityAdded(amountToken,amountCoin,liquidity) (Token.sol
             ↪ #199)
    Reentrancy in Rewardo_Token._transfer(address,address,uint256) (Token.
       ↪ sol#295-374):
          External calls:
          - _swapTokensForOtherTokens(token2Swap) (Token.sol#340)
                - routerV2.
                   ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
```

```
                    ↪ (tokenAmount,0,path,address(this),block.timestamp)
                    ↪ (Token.sol#138)
          - success = _sendInOtherTokens(marketingAddress,marketingPortion)
             ↪  (Token.sol#345)
                - feeToken.transfer(to,amount) (Token.sol#127)
          Event emitted after the call(s):
          - marketingFeeSent(marketingAddress,marketingPortion) (Token.sol
             ↪ #347)
   Reentrancy in Rewardo_Token._transfer(address,address,uint256) (Token.
      ↪ sol#295-374):
          External calls:
          - _swapTokensForOtherTokens(token2Swap) (Token.sol#340)
                - routerV2.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (tokenAmount,0,path,address(this),block.timestamp)
                    ↪ (Token.sol#138)
          - success = _sendInOtherTokens(marketingAddress,marketingPortion)
             ↪  (Token.sol#345)
                - feeToken.transfer(to,amount) (Token.sol#127)
          - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this
                    ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                    ↪ Token.sol#210)
                - routerV2.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ Token.sol#184)
          External calls sending eth:
          - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this
                    ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                    ↪ Token.sol#210)
          Event emitted after the call(s):
          - Approval(owner,spender,amount) (ERC20.sol#313)
```

```
                - _swapAndLiquify(_liquidityPending) (Token.sol#355)
        - liquidityAdded(amountToken,amountCoin,liquidity) (Token.sol
            ↪ #199)
                - _swapAndLiquify(_liquidityPending) (Token.sol#355)
Reentrancy in Rewardo_Token._transfer(address,address,uint256) (Token.
    ↪ sol#295-374):
        External calls:
        - _swapTokensForOtherTokens(token2Swap) (Token.sol#340)
                - routerV2.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (tokenAmount,0,path,address(this),block.timestamp)
                    ↪ (Token.sol#138)
        - success = _sendInOtherTokens(marketingAddress,marketingPortion)
            ↪  (Token.sol#345)
                - feeToken.transfer(to,amount) (Token.sol#127)
        - _swapAndLiquify(_liquidityPending) (Token.sol#355)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this
                    ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                    ↪ Token.sol#210)
                - routerV2.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ Token.sol#184)
        - _sendDividends(_rewardsPending) (Token.sol#360)
                - success = IERC20(rewardToken).approve(address(
                    ↪ dividendTracker),dividends) (Token.sol#249)
                - routerV2.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (tokenAmount,0,path,address(this),block.timestamp)
                    ↪ (Token.sol#240)
                - dividendTracker.distributeDividends(dividends) (Token.
                    ↪ sol#252)
        External calls sending eth:
        - _swapAndLiquify(_liquidityPending) (Token.sol#355)
```

```
              - routerV2.addLiquidityETH{value: coinAmount}(address(this
                  ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                  ↪ Token.sol#210)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (ERC20.sol#313)
              - _sendDividends(_rewardsPending) (Token.sol#360)
        - Transfer(from,to,amount) (ERC20.sol#237)
              - super._transfer(from,to,amount) (Token.sol#367)
        - rewardsFeeSent(dividends) (Token.sol#253)
              - _sendDividends(_rewardsPending) (Token.sol#360)
Reentrancy in Rewardo_Token._updateRouterV2(address) (Token.sol#376-387)
    ↪ :
        External calls:
        - pairV2 = IUniswapV2Factory(routerV2.factory()).createPair(
            ↪ address(this),routerV2.WETH()) (Token.sol#378)
        - _excludeFromDividends(router,true) (Token.sol#380)
              - dividendTracker.excludeFromDividends(account,balanceOf(
                  ↪ account),isExcluded) (Token.sol#263)
        Event emitted after the call(s):
        - ExcludeFromLimits(account,isExcluded) (Token.sol#415)
              - _excludeFromLimits(router,true) (Token.sol#382)
Reentrancy in Rewardo_Token._updateRouterV2(address) (Token.sol#376-387)
    ↪ :
        External calls:
        - pairV2 = IUniswapV2Factory(routerV2.factory()).createPair(
            ↪ address(this),routerV2.WETH()) (Token.sol#378)
        - _excludeFromDividends(router,true) (Token.sol#380)
              - dividendTracker.excludeFromDividends(account,balanceOf(
                  ↪ account),isExcluded) (Token.sol#263)
        - _setAMMPair(pairV2,true) (Token.sol#384)
              - dividendTracker.excludeFromDividends(account,balanceOf(
                  ↪ account),isExcluded) (Token.sol#263)
        Event emitted after the call(s):
        - AMMPairsUpdated(pair,isPair) (Token.sol#405)
```

```
                - _setAMMPair(pairV2,true) (Token.sol#384)
            - ExcludeFromLimits(account,isExcluded) (Token.sol#415)
                - _setAMMPair(pairV2,true) (Token.sol#384)
            - RouterV2Updated(router) (Token.sol#386)
    Reentrancy in DividendPayingToken._withdrawDividend(address) (
        ↪ TokenDividendTracker.sol#204-223):
            External calls:
            - IERC20(rewardToken).transfer(account,withdrawableDividend) (
                ↪ TokenDividendTracker.sol#210-219)
            Event emitted after the call(s):
            - DividendWithdrawn(account,withdrawableDividend) (
                ↪ TokenDividendTracker.sol#212)
    Reentrancy in Rewardo_Token.addLiquidityFromLeftoverTokens() (Token.sol
        ↪ #213-219):
            External calls:
            - unaddedTokens = _swapAndLiquify(leftoverTokens) (Token.sol#216)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this
                    ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                    ↪ Token.sol#210)
                - routerV2.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ Token.sol#184)
            External calls sending eth:
            - unaddedTokens = _swapAndLiquify(leftoverTokens) (Token.sol#216)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this
                    ↪ ),tokenAmount,0,0,address(0),block.timestamp) (
                    ↪ Token.sol#210)
            Event emitted after the call(s):
            - ForceLiquidityAdded(leftoverTokens,unaddedTokens) (Token.sol
                ↪ #218)
    Reentrancy in DividendPayingToken.distributeDividends(uint256) (
        ↪ TokenDividendTracker.sol#188-202):
            External calls:
```

```
        - IERC20(rewardToken).transferFrom(msg.sender,address(this),
            ↪ amount) (TokenDividendTracker.sol#192)
        Event emitted after the call(s):
        - DividendsDistributed(msg.sender,received) (TokenDividendTracker
            ↪ .sol#198)
Reentrancy in Rewardo_Token.initialize(address,address,address) (Token.
    ↪ sol#109-114):
        External calls:
        - _setRewardToken(_rewardToken) (Token.sol#111)
                - dividendTracker.setRewardToken(_rewardToken) (
                    ↪ TokenDividendTracker.sol#512)
        - _updateRouterV2(_router) (Token.sol#113)
                - dividendTracker.excludeFromDividends(account,balanceOf(
                    ↪ account),isExcluded) (Token.sol#263)
                - pairV2 = IUniswapV2Factory(routerV2.factory()).
                    ↪ createPair(address(this),routerV2.WETH()) (Token.
                    ↪ sol#378)
        Event emitted after the call(s):
        - AMMPairsUpdated(pair,isPair) (Token.sol#405)
                - _updateRouterV2(_router) (Token.sol#113)
        - ExcludeFromLimits(account,isExcluded) (Token.sol#415)
                - _updateRouterV2(_router) (Token.sol#113)
        - RouterV2Updated(router) (Token.sol#386)
                - _updateRouterV2(_router) (Token.sol#113)
Reentrancy in DividendTracker.process(uint256) (TokenDividendTracker.sol
    ↪ #455-492):
        External calls:
        - claim(account) (TokenDividendTracker.sol#475)
                - IERC20(rewardToken).transfer(account,
                    ↪ withdrawableDividend) (TokenDividendTracker.sol
                    ↪ #210-219)
        Event emitted after the call(s):
        - ProcessedDividendTracker(iterations,claims) (
            ↪ TokenDividendTracker.sol#491)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3
INFO:Detectors:
DividendTracker.getAccountData(address) (TokenDividendTracker.sol
    ↪ #376-405) uses timestamp for comparisons
        Dangerous comparisons:
        - nextClaimTime > block.timestamp (TokenDividendTracker.sol#404)
DividendTracker._canAutoClaim(uint256) (TokenDividendTracker.sol
    ↪ #435-439) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < lastClaimTime (TokenDividendTracker.sol#436)
        - block.timestamp - lastClaimTime >= claimWait (
            ↪ TokenDividendTracker.sol#438)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #block-timestamp
INFO:Detectors:
Different versions of Solidity are used:
        - Version used: ['0.8.19', '>=0.5.0', '>=0.6.2', '^0.8.0']
        - 0.8.19 (Token.sol#7)
        - >=0.5.0 (IUniswapV2Factory.sol#1)
        - >=0.5.0 (IUniswapV2Pair.sol#1)
        - >=0.6.2 (IUniswapV2Router01.sol#1)
        - >=0.6.2 (IUniswapV2Router02.sol#1)
        - ^0.8.0 (Context.sol#4)
        - ^0.8.0 (ERC20.sol#4)
        - ^0.8.0 (ERC20Burnable.sol#4)
        - ^0.8.0 (IERC20.sol#4)
        - ^0.8.0 (IERC20Metadata.sol#4)
        - ^0.8.0 (Initializable.sol#3)
        - ^0.8.0 (Ownable.sol#4)
        - ^0.8.0 (Ownable2Step.sol#4)
        - ^0.8.0 (TokenDividendTracker.sol#6)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used
```

```
INFO:Detectors:
Rewardo_Token._transfer(address,address,uint256) (Token.sol#295-374) has
    ↪ a high cyclomatic complexity (18).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #cyclomatic-complexity
INFO:Detectors:
Context._contextSuffixLength() (Context.sol#25-27) is never used and
    ↪ should be removed
Context._msgData() (Context.sol#21-23) is never used and should be
    ↪ removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code
INFO:Detectors:
Pragma version^0.8.0 (Context.sol#4) allows old versions
Pragma version^0.8.0 (ERC20.sol#4) allows old versions
Pragma version^0.8.0 (ERC20Burnable.sol#4) allows old versions
Pragma version^0.8.0 (IERC20.sol#4) allows old versions
Pragma version^0.8.0 (IERC20Metadata.sol#4) allows old versions
Pragma version>=0.5.0 (IUniswapV2Factory.sol#1) allows old versions
Pragma version>=0.5.0 (IUniswapV2Pair.sol#1) allows old versions
Pragma version>=0.6.2 (IUniswapV2Router01.sol#1) allows old versions
Pragma version>=0.6.2 (IUniswapV2Router02.sol#1) allows old versions
Pragma version^0.8.0 (Initializable.sol#3) allows old versions
Pragma version^0.8.0 (Ownable.sol#4) allows old versions
Pragma version^0.8.0 (Ownable2Step.sol#4) allows old versions
Pragma version0.8.19 (Token.sol#7) necessitates a version too recent to
    ↪ be trusted. Consider deploying with 0.8.18.
Pragma version^0.8.0 (TokenDividendTracker.sol#6) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (IUniswapV2Pair.sol#18) is
    ↪ not in mixedCase
```

```
Function IUniswapV2Pair.PERMIT_TYPEHASH() (IUniswapV2Pair.sol#19) is not
    ↪  in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (IUniswapV2Pair.sol#36) is
    ↪ not in mixedCase
Function IUniswapV2Router01.WETH() (IUniswapV2Router01.sol#5) is not in
    ↪ mixedCase
Contract Rewardo_Token (Token.sol#20-446) is not in CapWords
Event Rewardo_TokenmarketingAddressUpdated(address) (Token.sol#52) is
    ↪ not in CapWords
Event Rewardo_TokenmarketingFeesUpdated(uint16,uint16,uint16) (Token.sol
    ↪ #53) is not in CapWords
Event Rewardo_TokenmarketingFeeSent(address,uint256) (Token.sol#54) is
    ↪ not in CapWords
Event Rewardo_TokenliquidityFeesUpdated(uint16,uint16,uint16) (Token.sol
    ↪ #56) is not in CapWords
Event Rewardo_TokenliquidityAdded(uint256,uint256,uint256) (Token.sol
    ↪ #57) is not in CapWords
Event Rewardo_TokenrewardsFeesUpdated(uint16,uint16,uint16) (Token.sol
    ↪ #60) is not in CapWords
Event Rewardo_TokenrewardsFeeSent(uint256) (Token.sol#61) is not in
    ↪ CapWords
Parameter Rewardo_Token.initialize(address,address,address)._feeToken (
    ↪ Token.sol#109) is not in mixedCase
Parameter Rewardo_Token.initialize(address,address,address)._rewardToken
    ↪  (Token.sol#109) is not in mixedCase
Parameter Rewardo_Token.initialize(address,address,address)._router (
    ↪ Token.sol#109) is not in mixedCase
Parameter Rewardo_Token.updateSwapThreshold(uint16)._swapThresholdRatio
    ↪ (Token.sol#141) is not in mixedCase
Parameter Rewardo_Token.marketingAddressSetup(address)._newAddress (
    ↪ Token.sol#156) is not in mixedCase
Parameter Rewardo_Token.marketingFeesSetup(uint16,uint16,uint16)._buyFee
    ↪  (Token.sol#166) is not in mixedCase
```

```
Parameter Rewardo_Token.marketingFeesSetup(uint16,uint16,uint16).
    ↪ _sellFee (Token.sol#166) is not in mixedCase
Parameter Rewardo_Token.marketingFeesSetup(uint16,uint16,uint16).
    ↪ _transferFee (Token.sol#166) is not in mixedCase
Parameter Rewardo_Token.liquidityFeesSetup(uint16,uint16,uint16)._buyFee
    ↪  (Token.sol#221) is not in mixedCase
Parameter Rewardo_Token.liquidityFeesSetup(uint16,uint16,uint16).
    ↪ _sellFee (Token.sol#221) is not in mixedCase
Parameter Rewardo_Token.liquidityFeesSetup(uint16,uint16,uint16).
    ↪ _transferFee (Token.sol#221) is not in mixedCase
Parameter Rewardo_Token.rewardsFeesSetup(uint16,uint16,uint16)._buyFee (
    ↪ Token.sol#266) is not in mixedCase
Parameter Rewardo_Token.rewardsFeesSetup(uint16,uint16,uint16)._sellFee
    ↪ (Token.sol#266) is not in mixedCase
Parameter Rewardo_Token.rewardsFeesSetup(uint16,uint16,uint16).
    ↪ _transferFee (Token.sol#266) is not in mixedCase
Parameter Rewardo_Token.updateMaxWalletAmount(uint256)._maxWalletAmount
    ↪ (Token.sol#418) is not in mixedCase
Variable Rewardo_Token.AMMPairs (Token.sol#44) is not in mixedCase
Constant DividendPayingToken.magnitude (TokenDividendTracker.sol#175) is
    ↪  not in UPPER_CASE_WITH_UNDERSCORES
Parameter DividendTracker.setRewardToken(address)._rewardToken (
    ↪ TokenDividendTracker.sol#341) is not in mixedCase
Parameter DividendTracker.getAccountData(address)._account (
    ↪ TokenDividendTracker.sol#376) is not in mixedCase
Parameter DividendTrackerFunctions.gasForProcessingSetup(uint256).
    ↪ _gasForProcessing (TokenDividendTracker.sol#517) is not in
    ↪ mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256
    ↪ ,uint256,uint256,address,uint256).amountADesired (
    ↪ IUniswapV2Router01.sol#10) is too similar to IUniswapV2Router01.
    ↪ addLiquidity(address,address,uint256,uint256,uint256,uint256,
    ↪ address,uint256).amountBDesired (IUniswapV2Router01.sol#11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-too-similar
INFO:Detectors:
Rewardo_Token.constructor() (Token.sol#72-104) uses literals with too
    ↪ many digits:
        - gasForProcessingSetup(300000) (Token.sol#86)
Rewardo_Token.constructor() (Token.sol#72-104) uses literals with too
    ↪ many digits:
        - updateMaxWalletAmount(200000000 * (10 ** decimals()) / 10) (
            ↪ Token.sol#100)
Rewardo_Token.constructor() (Token.sol#72-104) uses literals with too
    ↪ many digits:
        - _mint(supplyRecipient,10000000000 * (10 ** decimals()) / 10) (
            ↪ Token.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #too-many-digits
INFO:Detectors:
DividendTracker.minimumTokenBalanceForDividends (TokenDividendTracker.
    ↪ sol#330) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-immutable
INFO:Slither:Token.sol analyzed (22 contracts with 85 detectors), 100
    ↪ result(s) found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

5 Conclusion

We examined the design and implementation of Rewardo Token in this audit and found several issues of various severities. We advise Rewardo Token team to implement the recommendations contained in all 3 of our findings to further enhance the code's security. It is of utmost priority to start by addressing the most severe exploit discovered by the auditors then followed by the remaining exploits, and finally we will be conducting a re-audit following the implementation of the remediation plan contained in this report.

We would much appreciate any constructive feedback or suggestions regarding our methodology, audit findings, or potential scope gaps in this report.

BLOCKHAT

SECURITY

For a Smart Contract Audit, contact us at contact@blockhat.io

Post-audit notes:

Non-Withdrawable Ether Generated swapAndLiquify Function: We evaluated this finding, and found that adding a Withdraw function would not do anything, since the contract has been renounced. We also analyzed that the amount of Ether being burned is so small, so that it will not affect liquidity that much as we go forward. Leaving as it is.

Use of Outdated ERC20 Implementation: We analyzed the potential security risks with this, and the only one we found was that it might be a little more costly to use gas. Other security risks has been cleared since the contract has been renounced. Leaving as is.

Unnecessary Override in _beforeTokenTransfer Function: This is related to the reward code in the transaction. It does not pose any harm, and contract has been renounced. Leaving as is.