QuillAudits

# Audit Report
# August, 2021

# Contents

## Scope of Audit

The scope of this audit was to analyze and document the DEFINE Token smart contract codebase for quality, security, and correctness.

## Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

# Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

**Structural Analysis**
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

**Static Analysis**
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

**Code Review / Manual Analysis**
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

**Gas Consumption**
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

**Tools and Platforms used for Audit**
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

# Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

## High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

## Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

## Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

## Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

# Number of issues per severity

## Define.sol

| Type | High | Medium | Low | Informational |
|---|---|---|---|---|
| Open | 0 | 0 | 0 | 0 |
| Acknowledged | 0 | 1 | 4 | 7 |
| Closed | 3 | 3 | 0 | 1 |

## DefineDividendTracker.sol

| Type | High | Medium | Low | Informational |
|---|---|---|---|---|
| Open | 0 | 0 | 0 | 0 |
| Acknowledged | 0 | 0 | 2 | 3 |
| Closed | 0 | 0 | 0 | 0 |

# Introduction

During the period of **July 23, 2021 to August 19** - QuillAudits Team performed a security audit for DEFINE Token smart contracts.

The code for the audit was taken from following the official link: https://github.com/playtimeprofit/definetics

| Note | Date | Commit hash |
|---|---|---|
| Version 1 | July 29 | 023ddbd2182d96306fb020573a64b50eaed0a87a |
| Version 2 | August 19 | 517febe6052669b30f3bf7993c49ac90d6e5545a |

# Issues Found – Code Review / Manual Testing

## A. Contract - Define.sol

### High severity issues

1. Business Logic Of The Transaction Fees

**Description**
As clarified by the Auditee, the fees will be charged on the receivers when:
- there is no in current moment swap (swapping lock)
- current time more than tradingEnabledTimestamp time
- the sender is not equal bounceFixedSaleWallet or receiver is owner

Therefore:
- 11% tax is charged on buys/sells and  ETH is redistributed to Definetics holders.
- 5% is charged and sent to owner's address

But during the test, we discovered that all 16% of the fees were sent to the Define.sol contract address by the following operation super._transfer(from, address(this), fees);. Moreover, no ETH was sent from the buyer or sellers to Definetics holders.

**Remediation**
We recommend changing the code or having further consideration on the business logic of the contract regarding the transaction fees.

**Status:** Fixed
The Auditee decided to add a condition to the _transfer() function, whereas when the contract balance has less than swapTokensAtAmount, all commision will be collected and sent to contract. Otherwise:
- 11% tax is charged on buys/sells and  ETH is redistributed to Definetics holders.
- 5% is charged and sent to owner's address.

## 2. Business Logic Of The swapAndSendToOwner() function

### Description

As clarified by the Auditee, the swapAndSendToOwner() aims to exchange DEFINE token to ETH and sends to the owner's wallet for trading DEFINE tokens.

However, during the test, we discovered the following issues:

1. The token is not exchanged to ETH before sending it to the owner's wallet.
2. The amount of tokens that are meant to be sent to the owner's wallet is calculated based on the balance of the Define.sol contract but the sender here is the msg.sender. Therefore, if the balance of the sender is smaller than contractTokenBalance.mul(liquidityFee).div(totalFees), the transaction will be reverted.
3. The newBalance is always Zero since the balance of the Define.sol contract does not get changed.

### Remediation

We recommend changing or having further consideration on the business logic for this function.

**Status:** Fixed

## 3. The _bounceFixedSaleWallet is not excluded when a new DividendTracker() is updated.

### Description

The _bounceFixedSaleWallet address is not excluded from dividends (excludeFromDividends) when the updateDividendTracker() is called.

### Remediation

Please help us confirm this implementation is in line with the business logic of the contract; otherwise, we recommend adding the _bounceFixedSaleWallet to the excludeFromDividends when the updateDividendTracker is called.

**Status:** Fixed

## Medium severity issues

### 4. Centralization Risks

#### Description

The role owner has the authority to update the critical settings:

- excludeFromFee
- updateClaimWait
- updateDividendTracker
- updateGasForProcess
- updateLiquidityWallet.

During the audit, we realized that the owner has complete control over the DividendTracker contract, which means that a malicious owner can replace the malicious DividendTracker address to obtain their own benefits.

#### Remediation

We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

- with reasonable latency for community awareness on privileged operations;
- Multisig with community-voted 3rd-party independent co-signers;
- DAO or Governance module increasing transparency and community involvement;

**Status: Acknowledged by the Auditee**

### 5. Low-level function calls

#### Description

The low-level function .call is being utilized in this contract, which can introduce several unexpected risks or errors.

**L493 - Define.sol:** (bool success,) = address(dividendTracker).call{value: dividends}("");

**L94 - DividendPayingToken.sol:** (bool success,) = user.call{value: _withdrawableDividend, gas: 3000}("");

As also mentioned in the “Centralization Risks" issue, the owner has complete control over the DividendTracker contract, which means that a malicious owner can replace the malicious DividendTracker address to obtain their own benefits. Therefore, the malicious contract of the dividendTracker will be able to reenter the Define.sol contract thanks to the .call function.

One of the major dangers of this implementation is that they can take over the control flow. In the reentrancy attack (a.k.a. recursive call attack), a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways.

**Remediation**
Even though we didn’t find any potential hacks via this low-level function for this contract. Nevertheless, please ensure that dividendTracker is a trusted contract and that it is thoroughly reviewed prior to use.

Moreover, you can follow the best practices to avoid Reentrancy weaknesses:

- Make sure all internal state changes are performed before the call is executed. This is known as the Checks-Effects-Interactions pattern
- Use a reentrancy lock (ie. OpenZeppelin's ReentrancyGuard).

**Status:** Fixed

## 6. Missing Check for Reentrancy Attack

**Description**
Calling define.transfer() might trigger the function uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens, which is implemented by a third party at uniswapV2Router, in swapTokensForEth. If there are vulnerable external calls in uniswapV2Router , reentrancy attacks could be conducted because this function has state updates and event emits after external calls.
The scope of the audit would treat the third-party implementation at uniswapV2Router as a black box and assume its functional correctness. However, third parties may be compromised in the real world that leads to assets lost or stolen.

**Remediation**
We recommend applying OpenZeppelin's ReentrancyGuard library - modifier for the aforementioned function to prevent reentrancy attack. Moreover, please assess the risks of third-party dependencies carefully.

**Status:** Fixed

### 7. Loops in the Contract are extremely costly

**Description**
The for loops in the entire codebase includes state variables .length of a non-memory array, in the condition of the for loops. As a result, these state variables consume a lot more extra gas for every iteration of the for loop.

**Remediation**
We recommend using a local variable instead of a state variable .length in a loop for the entire codebase.

**Status:** Fixed

## Low level severity issues

### 8. Dead code

**Description**
It was discovered that the function addLiquidity() is not used in this contract.
Unused code is allowed in Solidity and they do not pose a direct security issue. It is best practice though, to avoid them as they can:

- cause an increase in computations (and unnecessary gas consumption)
- indicate bugs or malformed data structures, and they are generally a sign of poor code quality
- cause code noise and decrease the readability of the code

**Remediation**
We recommend removing all unused variables/code from the codebase.

**Status:** Acknowledged by the Auditee

## 9. Missing zero address validation

### Description

We've detected missing zero address validation for the following functions:
constructor()
updateDividendTracker()
updateUniswapV2Router()
excludeFromFees()
excludeMultipleAccountsFromFees()
addFixedSaleEarlyParticipants()
setAutomatedMarketMakerPair()
updateLiquidityWallet()

### Remediation

Consider implementing require statements where appropriate to validate all user-controlled input, including constructor, to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

**Status: Acknowledged by the Auditee**

## 10. Floating Pragma

### Description

The Define.sol contract is being utilized with solidity ^0.8.0, the pragma version is floating. Therefore, locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

### Remediation

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

**Status: Acknowledged by the Auditee**

### 11. Avoid to use tx.origin

**L418:** emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas, tx.origin);

**Description**
tx.origin is a global variable in Solidity which returns the address of the account that sent the transaction. The contract does not use the variable for authorization but if this is used in the next development for authorization purpose, it could make a contract vulnerable if an authorized account calls into a malicious contract.

A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.

**Remediation**
tx.origin should not be used. We recommend using msg.sender instead.

**Status: Acknowledged by the Auditee**

## Informational

### 12. Removing irrelevant comments

**Description**
We discovered that there are many irrelevant comments in the swapAndSendToOwner() function. We recommend removing those comments to increase the readability and quality of the code.

**Status:** Fixed

### 13. Public function that could be declared external

**Description**
The following public functions that are never called by the contract should be declared external to save gas:

- updateDividendTracker()
- updateUniswapV2Router()

- excludeMultipleAccountsFromFees()
- setAutomatedMarketMakerPair()
- updateLiquidityWallet()
- updateGasForProcessing()

**Remediation**
Use the external attribute for functions never called from the contract.

**Status: Acknowledged by the Auditee**

### 14. block.timestamp may not be reliable

**Description**
The Define contract uses the block.timestamp as part of the calculations and time checks.

Nevertheless, timestamps can be slightly altered by miners to favor them in contracts that have logics that depend strongly on them. Consider taking into account this issue and warning the users that such a scenario could happen.

**Status: Acknowledged by the Auditee**

### 15. Immutable variables that could be declared constant

**Description**
Immutable state variables should be declared constant to save gas. The following immutable variables are not being re-initialized in the constructor():

- fixedSaleEarlyParticipantBuysThreshold
- fixedSaleEarlyParticipantDuration
- maxSellTransactionAmount
- sellFeeIncreaseFactor
- swapTokensAtAmount
- tradingEnabledTimestamp

Compared to regular state variables, the gas costs of constant and immutable variables are much lower. For a constant variable, the expression assigned to it is copied to all the places where it is accessed and also re-evaluated each time. This allows for local optimizations. Immutable variables are evaluated once at construction time and their value is copied to all the places in the code where they are accessed. For these values, 32 bytes are reserved, even if they would fit in fewer bytes. Due to this, constant values can sometimes be cheaper than immutable values.

**Remediation**
We recommend considering and changing the above-mentioned variables to constant.

**Status: Acknowledged by the Auditee**

### 16. _transfer() function should do one thing

**Description**
We recommend adding other functions for the _transfer function to increase the readability and quality of the code. The _transfer() function should just do the transfer rather than doing multiple operations such as:
- adding numberOfFixedSaleBuys
- updating dividendTracker balance
- processing gas

**Status: Acknowledged by the Auditee**

### 17. Inconsistent coding style

**Description**
Deviations from the Solidity Style Guide were identified throughout the entire codebase. Taking into consideration how much value a consistent coding style adds to the project's readability, enforcing a standard coding style with the help of linter tools such as Solhint is recommended.

**Status: Acknowledged by the Auditee**

## 18. Missing Range Check for Input Variable

### Description

To favor explicitness and readability, some functions and variables from the whole repository may benefit from a better naming.

Our suggestions are:
**L367:** canSwap to swapIsEnabled
**L379:** swapTokens to sendToOwnerAmount
**L382:** sellTokens to sendToDividendAmount
**L492:** dividends to defineContractBalance

Status: **Acknowledged by the Auditee**

## 19. Typos

### Description

Throughout the Define's codebase, there are a few typos in the code and in the comments. We list them here.

**L59:** // timestamp for when the token can be traded freely on PanackeSwap (PancakeSwap)
**L62:** // exlcude (exclude) from fees and max transaction amount

We recommend correcting the above-mentioned typos.

Status: **Acknowledged by the Auditee**

## B. Contract - DefineDividendTracker.sol

### High severity issues

No issues were found

### Medium level severity issues

No issues were found

### Low level severity issues

1. **nextClaimTime and withdrawableDividends are not checked**

   **Description**
   nextClaimTime and withdrawableDividends should be validated when the users want to claim by themselves. The check might help to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

   **Remediation**
   When the users want to claim by themselves, the nextClaimTime should be checked if it's valid and the balance of the withdrawableDividends of the users are available.

   **Status: Acknowledged by the Auditee**

2. **Floating Pragma**

   **Description**
   The Define.sol contract is being utilized with solidity ^0.8.0, the pragma version is floating. Therefore, locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

   **Remediation**
   Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile it locally.

**Status: Acknowledged by the Auditee**

## Informational

### 3. Conformance to Solidity naming conventions

**Description**
Constants should be named with all capital letters with underscores separating words. Examples: MAX_BLOCKS, TOKEN_NAME, TOKEN_TICKER, CONTRACT_VERSION.

Solidity defines a naming convention that should be followed. We recommend changing the magnitude to MAGNITUDE

**Status: Acknowledged by the Auditee**

### 4. Inconsistent coding style

**Description**
Deviations from the Solidity Style Guide were identified throughout the entire codebase. Taking into consideration how much value a consistent coding style adds to the project's readability, enforcing a standard coding style with the help of linter tools such as Solhint is recommended.

**Status: Acknowledged by the Auditee**

### 5. Missing docstrings

**Description**
It is extremely difficult to locate any contracts or functions, as they lack documentation. One consequence of this is that reviewers' understanding of the code's intention is impeded, which is significant because it is necessary to accurately determine both security and correctness.

They are additionally more readable and easier to maintain when wrapped in docstrings. The functions should be documented so that users can understand the purpose or intention of each function, as well as the situations in which it may fail, who is allowed to call it, what values it returns, and what events it emits.

**Remediation**

Consider thoroughly documenting all functions (and their parameters) that are part of the contracts' public API. Functions implementing sensitive functionality, even if those are not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

**Status: Acknowledged by the Auditee**

# Functional test - Define.sol

| Function Names | Testing results |
|---|---|
| constructor | Passed |
| updateDividendTracker | Passed |
| updateUniswapV2Router | Passed |
| excludeFromFees | Passed |
| excludeMultipleAccountsFromFees | Passed |
| addFixedSaleEarlyParticipants | Passed |
| setAutomatedMarketMakerPair | Passed |
| _setAutomatedMarketMakerPair | Passed |
| updateLiquidityWallet | Passed |
| updateGasForProcessing | Passed |
| updateClaimWait | Passed |
| getClaimWait | Passed |
| getTotalDividendsDistributed | Passed |
| isExcludedFromFees | Passed |
| withdrawableDividendOf | Passed |
| dividendTokenBalanceOf | Passed |
| getAccountDividendsInfo | Passed |
| getAccountDividendsInfoAtIndex | Passed |
| processDividendTracker | Passed |
| claim | Passed |

| Function Names | Testing results |
| --- | --- |
| getLastProcessedIndex | Passed |
| getNumberOfDividendTokenHolders | Passed |
| getTradingIsEnabled | Passed |
| _transfer (without taking fee) | Passed |
| _transfer (when taking fee) | Passed |
| swapAndSendToOwner | Passed |
| swapTokensForEth | Passed |
| addLiquidity | Passed |
| swapAndSendDividends | Passed |

# Functional test - DefineDividendTracker.sol

| Function Names | Testing results |
|---|---|
| distributeDividend | Passed |
| process | Passed |
| processAccount | Passed |
| transfer | Passed |
| transferFrom | Passed |
| updateClaimWait | Passed |
| withdrawDividend | Passed |
| approve | Passed |
| decreaseAllowance | Passed |
| excludeFromDividends | Passed |
| renounceOwnership | Passed |
| transferOwnership | Passed |

# Automated Testing - Define.sol

## Slither

```
INFO:Detectors:
Define.addLiquidity(uint256,uint256) (Define.sol#473-488) sends eth to arbitrary user
    Dangerous calls:
    - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,liquidityWallet,block.timestamp) (Define.sol#479-486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-
send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in Define._transfer(address,address,uint256) (Define.sol#312-424):
    External calls:
    - swapAndSendToOwner(swapTokens) (Define.sol#380)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,
path,address(this),block.timestamp) (Define.sol#463-469)
        - dividendTracker.setBalance(address(from),balanceOf(from)) (Define.sol#411)
        - dividendTracker.setBalance(address(to),balanceOf(to)) (Define.sol#412)
        - dividendTracker.process(gas) (Define.sol#417-422)
    - swapAndSendDividends(sellTokens) (Define.sol#383)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,
path,address(this),block.timestamp) (Define.sol#463-469)
    External calls sending eth:
    - swapAndSendToOwner(swapTokens) (Define.sol#380)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
    - swapAndSendDividends(sellTokens) (Define.sol#383)
        - (success) = address(dividendTracker).call{value: dividends}()
(Define.sol#493)
    State variables written after the call(s):
    - super._transfer(from,address(this),fees) (Define.sol#406)
        - _balances[sender] = senderBalance - amount (openzeppelin/contracts/token/ERC20/
ERC20.sol#232)
        - _balances[recipient] += amount (openzeppelin/contracts/token/ERC20/
ERC20.sol#234)
    - super._transfer(from,to,amount) (Define.sol#409)
        - _balances[sender] = senderBalance - amount (openzeppelin/contracts/token/ERC20/
ERC20.sol#232)
        - _balances[recipient] += amount (openzeppelin/contracts/token/ERC20/
ERC20.sol#234)
```

```
    - swapping = false (Define.sol#385)
Reentrancy in DividendPayingToken._withdrawDividendOfUser(address)
(DividendPayingToken.sol#89-105):
    External calls:
    - (success) = user.call{gas: 3000,value: _withdrawableDividend}()
(DividendPayingToken.sol#94)
    State variables written after the call(s):
    - withdrawnDividends[user] = withdrawnDividends[user].sub(_withdrawableDividend)
(DividendPayingToken.sol#97)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities
INFO:Detectors:
Define._transfer(address,address,uint256) (Define.sol#312-424) performs a multiplication on
the result of a division:
    -fees = amount.mul(totalFees).div(100) (Define.sol#397)
    -fees = fees.mul(sellFeeIncreaseFactor).div(100) (Define.sol#401)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply
INFO:Detectors:
Reentrancy in Define.updateDividendTracker(address) (Define.sol#163-178):
    External calls:
    - newDividendTracker.excludeFromDividends(address(newDividendTracker))
(Define.sol#170)
    - newDividendTracker.excludeFromDividends(address(this)) (Define.sol#171)
    - newDividendTracker.excludeFromDividends(owner()) (Define.sol#172)
    - newDividendTracker.excludeFromDividends(address(uniswapV2Router)) (Define.sol#173)
    State variables written after the call(s):
    - dividendTracker = newDividendTracker (Define.sol#177)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
INFO:Detectors:
Define._transfer(address,address,uint256).iterations (Define.sol#417) is a local variable never
initialized
Define._transfer(address,address,uint256).claims (Define.sol#417) is a local variable never
initialized
Define._transfer(address,address,uint256).lastProcessedIndex (Define.sol#417) is a local
variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-
variables
INFO:Detectors:
Define.claim() (Define.sol#296-298) ignores return value by
dividendTracker.processAccount(address(msg.sender),false) (Define.sol#297)
Define._transfer(address,address,uint256) (Define.sol#312-424) ignores return value by
dividendTracker.process(gas) (Define.sol#417-422)
Define.addLiquidity(uint256,uint256) (Define.sol#473-488) ignores return value by
uniswapV2Router.addLiquidityETH{value: ethAmount}
```

ethAmount}(address(this),tokenAmount,0,0,liquidityWallet,block.timestamp)
(Define.sol#479-486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
DividendPayingToken.constructor(string,string)._name (DividendPayingToken.sol#46)
shadows:
- ERC20._name (openzeppelin/contracts/token/ERC20/ERC20.sol#40) (state variable)
DividendPayingToken.constructor(string,string)._symbol (DividendPayingToken.sol#46)
shadows:
- ERC20._symbol (openzeppelin/contracts/token/ERC20/ERC20.sol#41) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Define.constructor(address)._uniswapV2Pair (Define.sol#126-127) lacks a zero-check on :
- uniswapV2Pair = _uniswapV2Pair (Define.sol#130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'Define._transfer(address,address,uint256).iterations (Define.sol#417)' in
Define._transfer(address,address,uint256) (Define.sol#312-424) potentially used before
declaration: ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin)
(Define.sol#418)
Variable 'Define._transfer(address,address,uint256).claims (Define.sol#417)' in
Define._transfer(address,address,uint256) (Define.sol#312-424) potentially used before
declaration: ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin)
(Define.sol#418)
Variable 'Define._transfer(address,address,uint256).lastProcessedIndex (Define.sol#417)' in
Define._transfer(address,address,uint256) (Define.sol#312-424) potentially used before
declaration: ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin)
(Define.sol#418)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in Define._transfer(address,address,uint256) (Define.sol#312-424):
External calls:
- swapAndSendToOwner(swapTokens) (Define.sol#380)
- (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,
path,address(this),block.timestamp) (Define.sol#463-469)
- dividendTracker.setBalance(address(from),balanceOf(from)) (Define.sol#411)
- dividendTracker.setBalance(address(to),balanceOf(to)) (Define.sol#412) -
dividendTracker.process(gas) (Define.sol#417-422)
- swapAndSendDividends(sellTokens) (Define.sol#383)
- (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
-

```
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,
path,address(this),block.timestamp) (Define.sol#463-469)
    External calls sending eth:
    - swapAndSendToOwner(swapTokens) (Define.sol#380)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
    - swapAndSendDividends(sellTokens) (Define.sol#383)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
    State variables written after the call(s):
    - swapAndSendDividends(sellTokens) (Define.sol#383)
        - _allowances[owner][spender] = amount (openzeppelin/contracts/token/ERC20/
ERC20.sol#311)
Reentrancy in Define.constructor(address) (Define.sol#110-157):
    External calls:
    - _uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.
WETH()) (Define.sol#126-127)
    State variables written after the call(s):
    - uniswapV2Pair = _uniswapV2Pair (Define.sol#130)
    - uniswapV2Router = _uniswapV2Router (Define.sol#129)
Reentrancy in Define.constructor(address) (Define.sol#110-157):
    External calls:
    - _uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.
WETH()) (Define.sol#126-127)
    - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (Define.sol#132)
        - dividendTracker.excludeFromDividends(pair) (Define.sol#220)
    State variables written after the call(s):
    - bounceFixedSaleWallet = _bounceFixedSaleWallet (Define.sol#135)
Reentrancy in Define.constructor(address) (Define.sol#110-157):
    External calls:
    - _uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.
WETH()) (Define.sol#126-127)
    - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (Define.sol#132)
        - dividendTracker.excludeFromDividends(pair) (Define.sol#220)
    - dividendTracker.excludeFromDividends(address(dividendTracker)) (Define.sol#138)
    - dividendTracker.excludeFromDividends(address(this)) (Define.sol#139)
    - dividendTracker.excludeFromDividends(owner()) (Define.sol#140)
    - dividendTracker.excludeFromDividends(address(_uniswapV2Router)) (Define.sol#141)
    - dividendTracker.excludeFromDividends(_bounceFixedSaleWallet) (Define.sol#142)
    State variables written after the call(s):
    - _mint(owner(),311622 * (10 ** 18)) (Define.sol#156)
        - _balances[account] += amount (openzeppelin/contracts/token/ERC20/ERC20.sol#256)
    - excludeFromFees(liquidityWallet,true) (Define.sol#145)
        - _isExcludedFromFees[account] = excluded (Define.sol#188)
    - excludeFromFees(address(this),true) (Define.sol#146)
```

```
        - _isExcludedFromFees[account] = excluded (Define.sol#188)
    - _mint(owner(),311622 * (10 ** 18)) (Define.sol#156)
        - _totalSupply += amount (openzeppelin/contracts/token/ERC20/ERC20.sol#255)
    - canTransferBeforeTradingIsEnabled[owner()] = true (Define.sol#149)
    - canTransferBeforeTradingIsEnabled[_bounceFixedSaleWallet] = true (Define.sol#150)
Reentrancy in DefineDividendTracker.processAccount(address,bool)
(DefineDividendTracker.sol#204-214):
    External calls:
    - amount = _withdrawDividendOfUser(account) (DefineDividendTracker.sol#205)
        - (success) = user.call{gas: 3000,value: _withdrawableDividend}()
(DividendPayingToken.sol#94)
    State variables written after the call(s):
    - lastClaimTimes[account] = block.timestamp (DefineDividendTracker.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2
INFO:Detectors:
Reentrancy in Define._setAutomatedMarketMakerPair(address,bool) (Define.sol#215-224):
    External calls:
    - dividendTracker.excludeFromDividends(pair) (Define.sol#220)
    Event emitted after the call(s):
    - SetAutomatedMarketMakerPair(pair,value) (Define.sol#223)
Reentrancy in Define._transfer(address,address,uint256) (Define.sol#312-424):
    External calls:
    - swapAndSendToOwner(swapTokens) (Define.sol#380)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,
path,address(this),block.timestamp) (Define.sol#463-469)
        - dividendTracker.setBalance(address(from),balanceOf(from)) (Define.sol#411)
        - dividendTracker.setBalance(address(to),balanceOf(to)) (Define.sol#412)
        - dividendTracker.process(gas) (Define.sol#417-422)
    - swapAndSendDividends(sellTokens) (Define.sol#383)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,
path,address(this),block.timestamp) (Define.sol#463-469)
    External calls sending eth:
- swapAndSendToOwner(swapTokens) (Define.sol#380)
    - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
- swapAndSendDividends(sellTokens) (Define.sol#383)
    - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
Event emitted after the call(s):
- Approval(owner,spender,amount) (openzeppelin/contracts/token/ERC20/ERC20.sol#312)
    - swapAndSendDividends(sellTokens) (Define.sol#383)
- SendDividends(tokens,dividends) (Define.sol#496)
    - swapAndSendDividends(sellTokens) (Define.sol#383)
```

```
    - Transfer(sender,recipient,amount) (openzeppelin/contracts/token/ERC20/
ERC20.sol#236)
        - super._transfer(from,to,amount) (Define.sol#409)
    - Transfer(sender,recipient,amount) (openzeppelin/contracts/token/ERC20/
ERC20.sol#236)
        - super._transfer(from,address(this),fees) (Define.sol#406)
Reentrancy in Define._transfer(address,address,uint256) (Define.sol#312-424):
    External calls:
    - swapAndSendToOwner(swapTokens) (Define.sol#380)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,
path,address(this),block.timestamp) (Define.sol#463-469)
        - dividendTracker.setBalance(address(from),balanceOf(from)) (Define.sol#411)
        - dividendTracker.setBalance(address(to),balanceOf(to)) (Define.sol#412)
        - dividendTracker.process(gas) (Define.sol#417-422)
    - swapAndSendDividends(sellTokens) (Define.sol#383)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,
path,address(this),block.timestamp) (Define.sol#463-469)
    - dividendTracker.setBalance(address(from),balanceOf(from)) (Define.sol#411)
    - dividendTracker.setBalance(address(to),balanceOf(to)) (Define.sol#412)
    - dividendTracker.process(gas) (Define.sol#417-422)
    External calls sending eth:
    - swapAndSendToOwner(swapTokens) (Define.sol#380)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
    - swapAndSendDividends(sellTokens) (Define.sol#383)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
    Event emitted after the call(s):
    - ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin)
(Define.sol#418)
Reentrancy in Define.constructor(address) (Define.sol#110-157):
    External calls:
    - _uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.
WETH()) (Define.sol#126-127)
    - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (Define.sol#132)
        - dividendTracker.excludeFromDividends(pair) (Define.sol#220)
    Event emitted after the call(s):
    - SetAutomatedMarketMakerPair(pair,value) (Define.sol#223)
        - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (Define.sol#132)
Reentrancy in Define.constructor(address) (Define.sol#110-157):
    External calls:
    - _uniswapV2Pair =
```

IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Define.sol#126-127)
- _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (Define.sol#132)
  - dividendTracker.excludeFromDividends(pair) (Define.sol#220)
- dividendTracker.excludeFromDividends(address(dividendTracker)) (Define.sol#138)
- dividendTracker.excludeFromDividends(address(this)) (Define.sol#139)
- dividendTracker.excludeFromDividends(owner()) (Define.sol#140)
- dividendTracker.excludeFromDividends(address(_uniswapV2Router)) (Define.sol#141)
- dividendTracker.excludeFromDividends(_bounceFixedSaleWallet) (Define.sol#142)

Event emitted after the call(s):
- ExcludeFromFees(account,excluded) (Define.sol#190)
  - excludeFromFees(liquidityWallet,true) (Define.sol#145)
- ExcludeFromFees(account,excluded) (Define.sol#190)
  - excludeFromFees(address(this),true) (Define.sol#146)
- Transfer(address(0),account,amount) (openzeppelin/contracts/token/ERC20/ERC20.sol#257)
  - _mint(owner(),311622 * (10 ** 18)) (Define.sol#156)

Reentrancy in DefineDividendTracker.processAccount(address,bool) (DefineDividendTracker.sol#204-214):

External calls:
- amount = _withdrawDividendOfUser(account) (DefineDividendTracker.sol#205)
  - (success) = user.call{gas: 3000,value: _withdrawableDividend}() (DividendPayingToken.sol#94)

Event emitted after the call(s):
- Claim(account,amount,automatic) (DefineDividendTracker.sol#209)

Reentrancy in Define.processDividendTracker(uint256) (Define.sol#291-294):

External calls:
- (iterations,claims,lastProcessedIndex) = dividendTracker.process(gas) (Define.sol#292)

Event emitted after the call(s):
- ProcessedDividendTracker(iterations,claims,lastProcessedIndex,false,gas,tx.origin) (Define.sol#293)

Reentrancy in Define.swapAndSendDividends(uint256) (Define.sol#490-498):

External calls:
- swapTokensForEth(tokens) (Define.sol#491)
  - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Define.sol#463-469)
- (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)

External calls sending eth:
- (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)

Event emitted after the call(s):
- SendDividends(tokens,dividends) (Define.sol#496)

Reentrancy in Define.swapAndSendToOwner(uint256) (Define.sol#426-450):

External calls:
- super.transfer(owner(),tokens) (Define.sol#447)

```
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,
path,address(this),block.timestamp) (Define.sol#463-469)
        - dividendTracker.setBalance(address(from),balanceOf(from)) (Define.sol#411)
        - dividendTracker.setBalance(address(to),balanceOf(to)) (Define.sol#412)
        - dividendTracker.process(gas) (Define.sol#417-422)
    External calls sending eth:
    - super.transfer(owner(),tokens) (Define.sol#447)
        - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
    Event emitted after the call(s):
    - SendToOwner(tokens,newBalance) (Define.sol#449)
Reentrancy in Define.updateDividendTracker(address) (Define.sol#163-178):
    External calls:
    - newDividendTracker.excludeFromDividends(address(newDividendTracker))
(Define.sol#170)
    - newDividendTracker.excludeFromDividends(address(this)) (Define.sol#171)
    - newDividendTracker.excludeFromDividends(owner()) (Define.sol#172)
    - newDividendTracker.excludeFromDividends(address(uniswapV2Router)) (Define.sol#173)
    Event emitted after the call(s):
    - UpdateDividendTracker(newAddress,address(dividendTracker)) (Define.sol#175)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
INFO:Detectors:
Define.getTradingIsEnabled() (Define.sol#308-310) uses timestamp for comparisons
    Dangerous comparisons:
    - block.timestamp >= tradingEnabledTimestamp (Define.sol#309)
Define._transfer(address,address,uint256) (Define.sol#312-424) uses timestamp for
comparisons
    Dangerous comparisons:
    - require(bool,string)(block.timestamp >= fixedSaleStartTimestamp,Define: The fixed-sale
has not started yet.) (Define.sol#338)
    - openToEveryone = block.timestamp.sub(fixedSaleStartTimestamp) >=
fixedSaleEarlyParticipantDuration || numberOfFixedSaleBuys >=
fixedSaleEarlyParticipantBuysThreshold (Define.sol#340-341)
    - ! swapping && tradingIsEnabled && automatedMarketMakerPairs[to] && from !=
address(uniswapV2Router) && ! _isExcludedFromFees[to] (Define.sol#356-360)
    - tradingIsEnabled && canSwap && ! swapping && ! automatedMarketMakerPairs[from] &&
from != liquidityWallet && to != liquidityWallet (Define.sol#370-375)
    - takeFee = ! isFixedSaleBuy && tradingIsEnabled && ! swapping (Define.sol#389)
DefineDividendTracker.getAccount(address) (DefineDividendTracker.sol#68-111) uses
timestamp for comparisons
    Dangerous comparisons:
    - nextClaimTime > block.timestamp (DefineDividendTracker.sol#108-110)
DefineDividendTracker.canAutoClaim(uint256) (DefineDividendTracker.sol#132-138) uses
timestamp for comparison
```

```
    Dangerous comparisons:
    - lastClaimTime > block.timestamp (DefineDividendTracker.sol#133)
    - block.timestamp.sub(lastClaimTime) >= claimWait (DefineDividendTracker.sol#137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (openzeppelin/contracts/utils/Context.sol#20-22) is never used and
should be removed
Define.addLiquidity(uint256,uint256) (Define.sol#473-488) is never used and should be
removed
DividendPayingToken._transfer(address,address,uint256) (DividendPayingToken.sol#145-151)
is never used and should be removed
SafeMath.div(uint256,uint256,string) (openzeppelin/contracts/utils/math/
SafeMath.sol#190-199) is never used and should be removed
SafeMath.mod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#150-152) is
never used and should be removed
SafeMath.mod(uint256,uint256,string) (openzeppelin/contracts/utils/math/
SafeMath.sol#216-225) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (openzeppelin/contracts/utils/math/
SafeMath.sol#167-176) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#21-27) is
never used and should be removed
SafeMath.tryDiv(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#63-68) is
never used and should be removed
SafeMath.tryMod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#75-80)
is never used and should be removed
SafeMath.tryMul(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#46-56) is
never used and should be removed
SafeMath.trySub(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#34-39) is
never used and should be removed
SafeMathInt.abs(int256) (libraries/SafeMathInt.sol#82-85) is never used and should be
removed
SafeMathInt.div(int256,int256) (libraries/SafeMathInt.sol#53-59) is never used and should be
removed
SafeMathInt.mul(int256,int256) (libraries/SafeMathInt.sol#41-48) 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 (Define.sol#1) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (DefineDividendTracker.sol#1) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (DividendPayingToken.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IDividendPayingToken.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6
```

```
Pragma version^0.8.0 (interfaces/IDividendPayingTokenOptional.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IUniswapV2Factory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IUniswapV2Router.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (libraries/IterableMapping.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (libraries/SafeMathInt.sol#28) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (libraries/SafeMathUint.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/math/SafeMath.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Define.swapAndSendDividends(uint256) (Define.sol#490-498):
    - (success) = address(dividendTracker).call{value: dividends}() (Define.sol#493)
Low level call in DividendPayingToken._withdrawDividendOfUser(address) (DividendPayingToken.sol#89-105):
    - (success) = user.call{gas: 3000,value: _withdrawableDividend}() (DividendPayingToken.sol#94)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable Define.ETHRewardsFee (Define.sol#28) is not in mixedCase
Parameter DefineDividendTracker.getAccount(address)._account (DefineDividendTracker.sol#68) is not in mixedCase
Parameter DividendPayingToken.dividendOf(address)._owner (DividendPayingToken.sol#111) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (DividendPayingToken.sol#118) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (DividendPayingToken.sol#125) is not in mixedCase
```

```
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner
(DividendPayingToken.sol#135) is not in mixedCase
Constant DividendPayingToken.magnitude (DividendPayingToken.sol#26) is not in
UPPER_CASE_WITH_UNDERSCORES
Function IUniswapV2Router01.WETH() (interfaces/IUniswapV2Router.sol#7) is not in
mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Detectors:
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend
(DividendPayingToken.sol#90) is too similar to
DefineDividendTracker.getAccount(address).withdrawableDividends
(DefineDividendTracker.sol#73)
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uin
t256).amountADesired (interfaces/IUniswapV2Router.sol#12) is too similar to
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uin
t256).amountBDesired (interfaces/IUniswapV2Router.sol#13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar
INFO:Detectors:
Define.updateGasForProcessing(uint256) (Define.sol#234-239) uses literals with too many
digits:
    - require(bool,string)(newValue >= 200000 && newValue <= 500000,Define:
gasForProcessing must be between 200,000 and 500,000) (Define.sol#235)
Define.slitherConstructorVariables() (Define.sol#11-499) uses literals with too many digits:
    - gasForProcessing = 300000 (Define.sol#36)
DefineDividendTracker.getAccountAtIndex(uint256) (DefineDividendTracker.sol#113-130) uses
literals with too many digits:
    - (0x0000000000000000000000000000000000000000,- 1,- 1,0,0,0,0,0)
(DefineDividendTracker.sol#124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
SafeMathInt.MAX_INT256 (libraries/SafeMathInt.sol#36) is never used in SafeMathInt
(libraries/SafeMathInt.sol#34-93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-
variables
INFO:Detectors:
Define.fixedSaleEarlyParticipantBuysThreshold (Define.sol#47) should be constant
Define.fixedSaleEarlyParticipantDuration (Define.sol#46) should be constant
Define.fixedSaleStartTimestamp (Define.sol#42) should be constant
Define.maxSellTransactionAmount (Define.sol#25) should be constant
Define.sellFeeIncreaseFactor (Define.sol#33) should be constant
Define.swapTokensAtAmount (Define.sol#26) should be constant
Define.tradingEnabledTimestamp (Define.sol#60) should be constant
```

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
updateDividendTracker(address) should be declared external:
    - Define.updateDividendTracker(address) (Define.sol#163-178)
updateUniswapV2Router(address) should be declared external:
    - Define.updateUniswapV2Router(address) (Define.sol#180-184)
excludeMultipleAccountsFromFees(address[],bool) should be declared external:
    - Define.excludeMultipleAccountsFromFees(address[],bool) (Define.sol#193-199)
setAutomatedMarketMakerPair(address,bool) should be declared external:
    - Define.setAutomatedMarketMakerPair(address,bool) (Define.sol#209-213)
updateLiquidityWallet(address) should be declared external:
    - Define.updateLiquidityWallet(address) (Define.sol#227-232)
updateGasForProcessing(uint256) should be declared external:
    - Define.updateGasForProcessing(uint256) (Define.sol#234-239)
isExcludedFromFees(address) should be declared external:
    - Define.isExcludedFromFees(address) (Define.sol#253-255)
withdrawableDividendOf(address) should be declared external:
    - Define.withdrawableDividendOf(address) (Define.sol#257-259)
dividendTokenBalanceOf(address) should be declared external:
    - Define.dividendTokenBalanceOf(address) (Define.sol#261-263)
withdrawDividend() should be declared external:
    - DefineDividendTracker.withdrawDividend() (DefineDividendTracker.sol#37-39)
    - DividendPayingToken.withdrawDividend() (DividendPayingToken.sol#83-85)
getAccountAtIndex(uint256) should be declared external:
    - DefineDividendTracker.getAccountAtIndex(uint256) (DefineDividendTracker.sol#113-130)
process(uint256) should be declared external:
    - DefineDividendTracker.process(uint256) (DefineDividendTracker.sol#157-202)
dividendOf(address) should be declared external:
    - DividendPayingToken.dividendOf(address) (DividendPayingToken.sol#111-113)
withdrawnDividendOf(address) should be declared external:
    - DividendPayingToken.withdrawnDividendOf(address) (DividendPayingToken.sol#125-127)
get(IterableMapping.Map,address) should be declared external:
    - IterableMapping.get(IterableMapping.Map,address) (libraries/IterableMapping.sol#13-15)
getIndexOfKey(IterableMapping.Map,address) should be declared external:
    - IterableMapping.getIndexOfKey(IterableMapping.Map,address) (libraries/IterableMapping.sol#17-22)
getKeyAtIndex(IterableMapping.Map,uint256) should be declared external:
    - IterableMapping.getKeyAtIndex(IterableMapping.Map,uint256) (libraries/IterableMapping.sol#24-26)
size(IterableMapping.Map) should be declared external:
    - IterableMapping.size(IterableMapping.Map) (libraries/IterableMapping.sol#30-32)
renounceOwnership() should be declared external:
    - Ownable.renounceOwnership() (openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address)

(openzeppelin/contracts/access/Ownable.sol#61-64)
name() should be declared external:
- ERC20.name() (openzeppelin/contracts/token/ERC20/ERC20.sol#60-62)
symbol() should be declared external:
- ERC20.symbol() (openzeppelin/contracts/token/ERC20/ERC20.sol#68-70)
decimals() should be declared external:
- ERC20.decimals() (openzeppelin/contracts/token/ERC20/ERC20.sol#85-87)
allowance(address,address) should be declared external:
- ERC20.allowance(address,address) (openzeppelin/contracts/token/ERC20/ERC20.sol#119-121)
approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#130-133)
transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#148-162)
increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#195-203)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

## SOLIDITY STATIC ANALYSIS

Transaction origin:Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.morePos: 293:84:

Transaction origin:Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.morePos: 418:88:

Check-effects-interaction:Potential violation of Checks-Effects-Interaction pattern in Define.(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.morePos: 110:4:

Check-effects-interaction:Potential violation of Checks-Effects-Interaction pattern in Define.updateDividendTracker(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.morePos: 163:4:

Check-effects-interaction:Potential violation of Checks-Effects-Interaction pattern in Define._setAutomatedMarketMakerPair(address,bool): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.morePos: 215:4:
Check-effects-interaction:Potential violation of Checks-Effects-Interaction pattern in Define.processDividendTracker(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.morePos: 291:1:

Check-effects-interaction:Potential violation of Checks-Effects-Interaction pattern in Define._transfer(address,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.morePos: 312:4:

Check-effects-interaction:Potential violation of Checks-Effects-Interaction pattern in Define.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.morePos: 452:4:

Check-effects-interaction:Potential violation of Checks-Effects-Interaction pattern in Define.swapAndSendDividends(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.morePos: 490:4:

Check-effects-interaction:Potential violation of Checks-Effects-Interaction pattern in DividendPayingToken._withdrawDividendOfUser(address payable): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.morePos: 89:2:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 309:15:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 338:20:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 340:34:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 468:12:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 485:12:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 109:57:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 110:70:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 134:24:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 138:12:

Block timestamp:Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.morePos: 209:32:

Low level calls:Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.morePos: 493:26:

Low level calls:Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.morePos: 94:24:

## SOLHINT LINTER

```
Define.sol
   1:1   error    Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement  compiler-
version
  28:5   warning  Variable name must be in mixedCase                          var-name-mixedcase
 111:9   warning  Variable name must be in mixedCase                          var-name-mixedcase
 159:32  warning  Code contains empty blocks                                  no-empty-blocks
 164:9   warning  Error message for require is too long                       reason-string
 168:9   warning  Error message for require is too long                       reason-string
 181:9   warning  Error message for require is too long                       reason-string
 187:9   warning  Error message for require is too long                       reason-string
 210:9   warning  Error message for require is too long                       reason-string
 216:9   warning  Error message for require is too long                       reason-string
 228:9   warning  Error message for require is too long                       reason-string
 235:9   warning  Error message for require is too long                       reason-string
 236:9   warning  Error message for require is too long                       reason-string
 293:85  warning  Avoid to use tx.origin                                      avoid-tx-origin
 309:16  warning  Avoid to make time-based decisions in your business logic   not-rely-on-
time
 317:9   warning  Error message for require is too long                       reason-string
 318:9   warning  Error message for require is too long                       reason-string
 325:13  warning  Error message for require is too long                       reason-string
 338:13  warning  Error message for require is too long                       reason-string
 338:21  warning  Avoid to make time-based decisions in your business logic   not-rely-on-
time
 340:35  warning  Avoid to make time-based decisions in your business logic   not-rely-
on-time
 344:17  warning  Error message for require is too long                       reason-string
 362:13  warning  Error message for require is too long                       reason-string
 411:72  warning  Code contains empty blocks                                  no-empty-blocks
 411:81  warning  Code contains empty blocks                                  no-empty-blocks
 412:68  warning  Code contains empty blocks                                  no-empty-blocks
 412:77  warning  Code contains empty blocks                                  no-empty-blocks
 418:89  warning  Avoid to use tx.origin                                      avoid-tx-origin
 420:13  warning  Code contains empty blocks                                  no-empty-blocks
 468:13  warning  Avoid to make time-based decisions in your business logic   not-rely-on-
time
 485:13  warning  Avoid to make time-based decisions in your business logic   not-rely-on-
time
 493:27  warning  Avoid to use low level calls                                avoid-low-level-calls
```

### Results

The major issues were found, analyzed and reported. Some false positive errors were also reported by the tools. All the other issues have been categorized above according to their level of severity.

# Automated Testing - DefineDividendTracker.sol

## Slither

```
Reentrancy in DividendPayingToken._withdrawDividendOfUser(address)
(DividendPayingToken.sol#89-105):
    External calls:
    - (success) = user.call{gas: 3000,value: _withdrawableDividend}()
(DividendPayingToken.sol#94)
    State variables written after the call(s):
    - withdrawnDividends[user] = withdrawnDividends[user].sub(_withdrawableDividend)
(DividendPayingToken.sol#97)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities
INFO:Detectors:
DividendPayingToken.constructor(string,string)._name (DividendPayingToken.sol#46)
shadows:
    - ERC20._name (openzeppelin/contracts/token/ERC20/ERC20.sol#40) (state variable)
DividendPayingToken.constructor(string,string)._symbol (DividendPayingToken.sol#46)
shadows:
    - ERC20._symbol (openzeppelin/contracts/token/ERC20/ERC20.sol#41) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing
INFO:Detectors:
Reentrancy in DefineDividendTracker.processAccount(address,bool)
(DefineDividendTracker.sol#204-214):
    External calls:
    - amount = _withdrawDividendOfUser(account) (DefineDividendTracker.sol#205)
        - (success) = user.call{gas: 3000,value: _withdrawableDividend}()
(DividendPayingToken.sol#94)
    State variables written after the call(s):
    - lastClaimTimes[account] = block.timestamp (DefineDividendTracker.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2
INFO:Detectors:
Reentrancy in DefineDividendTracker.processAccount(address,bool)
(DefineDividendTracker.sol#204-214):
    External calls:
    - amount = _withdrawDividendOfUser(account) (DefineDividendTracker.sol#205)
        - (success) = user.call{gas: 3000,value: _withdrawableDividend}()
(DividendPayingToken.sol#94)
    Event emitted after the call(s):
    - Claim(account,amount,automatic) (DefineDividendTracker.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
```

```
INFO:Detectors:
DefineDividendTracker.getAccount(address) (DefineDividendTracker.sol#68-111) uses
timestamp for comparisons
    Dangerous comparisons:
    - nextClaimTime > block.timestamp (DefineDividendTracker.sol#108-110)
DefineDividendTracker.canAutoClaim(uint256) (DefineDividendTracker.sol#132-138) uses
timestamp for comparisons
    Dangerous comparisons:
    - lastClaimTime > block.timestamp (DefineDividendTracker.sol#133)
    - block.timestamp.sub(lastClaimTime) >= claimWait (DefineDividendTracker.sol#137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (openzeppelin/contracts/utils/Context.sol#20-22) is never used and
should be removed
DividendPayingToken._transfer(address,address,uint256) (DividendPayingToken.sol#145-151)
is never used and should be removed
ERC20._transfer(address,address,uint256) (openzeppelin/contracts/token/ERC20/
ERC20.sol#219-239) is never used and should be removed
SafeMath.div(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#134-136) is
never used and should be removed
SafeMath.div(uint256,uint256,string) (openzeppelin/contracts/utils/math/
SafeMath.sol#190-199) is never used and should be removed
SafeMath.mod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#150-152) is
never used and should be removed
SafeMath.mod(uint256,uint256,string) (openzeppelin/contracts/utils/math/
SafeMath.sol#216-225) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (openzeppelin/contracts/utils/math/
SafeMath.sol#167-176) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#21-27) is
never used and should be removed
SafeMath.tryDiv(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#63-68) is
never used and should be removed
SafeMath.tryMod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#75-80)
is never used and should be removed
SafeMath.tryMul(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#46-56) is
never used and should be removed
SafeMath.trySub(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#34-39) is
never used and should be removed
SafeMathInt.abs(int256) (libraries/SafeMathInt.sol#82-85) is never used and should be
removed
SafeMathInt.div(int256,int256) (libraries/SafeMathInt.sol#53-59) is never used and should be
removed
SafeMathInt.mul(int256,int256) (libraries/SafeMathInt.sol#41-48) 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 (DefineDividendTracker.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (DividendPayingToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IDividendPayingToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IDividendPayingTokenOptional.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (libraries/IterableMapping.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (libraries/SafeMathInt.sol#28) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (libraries/SafeMathUint.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/math/SafeMath.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in DividendPayingToken._withdrawDividendOfUser(address) (DividendPayingToken.sol#89-105):
	- (success) = user.call{gas: 3000,value: _withdrawableDividend}() (DividendPayingToken.sol#94)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter DefineDividendTracker.getAccount(address)._account (DefineDividendTracker.sol#68) is not in mixedCase
Parameter DividendPayingToken.dividendOf(address)._owner (DividendPayingToken.sol#111) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (DividendPayingToken.sol#118) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (DividendPayingToken.sol#125) is not in mixedCase

Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (DividendPayingToken.sol#135) is not in mixedCase
Constant DividendPayingToken.magnitude (DividendPayingToken.sol#26) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (DividendPayingToken.sol#90) is too similar to DefineDividendTracker.getAccount(address).withdrawableDividends (DefineDividendTracker.sol#73)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
DefineDividendTracker.getAccountAtIndex(uint256) (DefineDividendTracker.sol#113-130) uses literals with too many digits:
    - (0x0000000000000000000000000000000000000000,- 1,- 1,0,0,0,0,0) (DefineDividendTracker.sol#124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
SafeMathInt.MAX_INT256 (libraries/SafeMathInt.sol#36) is never used in SafeMathInt (libraries/SafeMathInt.sol#34-93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
withdrawDividend() should be declared external:
    - DefineDividendTracker.withdrawDividend() (DefineDividendTracker.sol#37-39)
    - DividendPayingToken.withdrawDividend() (DividendPayingToken.sol#83-85)
getAccountAtIndex(uint256) should be declared external:
    - DefineDividendTracker.getAccountAtIndex(uint256) (DefineDividendTracker.sol#113-130)
process(uint256) should be declared external:
    - DefineDividendTracker.process(uint256) (DefineDividendTracker.sol#157-202)
dividendOf(address) should be declared external:
    - DividendPayingToken.dividendOf(address) (DividendPayingToken.sol#111-113)
withdrawnDividendOf(address) should be declared external:
    - DividendPayingToken.withdrawnDividendOf(address) (DividendPayingToken.sol#125-127)
get(IterableMapping.Map,address) should be declared external:
    - IterableMapping.get(IterableMapping.Map,address) (libraries/IterableMapping.sol#13-15)
getIndexOfKey(IterableMapping.Map,address) should be declared external:
    - IterableMapping.getIndexOfKey(IterableMapping.Map,address) (libraries/IterableMapping.sol#17-22)
getKeyAtIndex(IterableMapping.Map,uint256) should be declared external:
    - IterableMapping.getKeyAtIndex(IterableMapping.Map,uint256) (libraries/IterableMapping.sol#24-26)
size(IterableMapping.Map) should be declared external:
    - IterableMapping.size(IterableMapping.Map) (libraries/IterableMapping.sol#30-32)

renounceOwnership() should be declared external:
    - Ownable.renounceOwnership() (openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address) (openzeppelin/contracts/access/Ownable.sol#61-64)
name() should be declared external:
    - ERC20.name() (openzeppelin/contracts/token/ERC20/ERC20.sol#60-62)
symbol() should be declared external:
    - ERC20.symbol() (openzeppelin/contracts/token/ERC20/ERC20.sol#68-70)
decimals() should be declared external:
    - ERC20.decimals() (openzeppelin/contracts/token/ERC20/ERC20.sol#85-87)
transfer(address,uint256) should be declared external:
    - ERC20.transfer(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#111-114)
allowance(address,address) should be declared external:
    - ERC20.allowance(address,address) (openzeppelin/contracts/token/ERC20/ERC20.sol#119-121)
approve(address,uint256) should be declared external:
    - ERC20.approve(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#130-133)
transferFrom(address,address,uint256) should be declared external:
    - ERC20.transferFrom(address,address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#148-162)
increaseAllowance(address,uint256) should be declared external:
    - ERC20.increaseAllowance(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
    - ERC20.decreaseAllowance(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#195-203)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

## SOLIDITY STATIC ANALYSIS

Check-effects-interaction:
Potential violation of Checks-Effects-Interaction pattern in DividendPayingToken._withdrawDividendOfUser(address payable): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 89:2:

Block timestamp:
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 109:57:

Block timestamp:
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 110:70:

Block timestamp:
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 134:24:

Block timestamp:
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 138:12:

Block timestamp:
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 209:32:

Low level calls:
Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 94:24:

## SOLHINT LINTER

```
DefineDividendTracker.sol
   1:1   error    Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement
compiler-version
  34:9   warning  Error message for require is too long                    reason-string
  38:9   warning  Error message for require is too long                    reason-string
  42:6   warning  Provide an error message for require                      reason-string
  52:9   warning  Error message for require is too long                    reason-string
  53:9   warning  Error message for require is too long                    reason-string
 108:58  warning  Avoid to make time-based decisions in your business logic  not-rely-
on-time
 109:71  warning  Avoid to make time-based decisions in your business logic  not-rely-
on-time
 133:25  warning  Avoid to make time-based decisions in your business logic  not-rely-
on-time
 137:13  warning  Avoid to make time-based decisions in your business logic  not-rely-
on-time
 208:33  warning  Avoid to make time-based decisions in your business logic  not-rely-
on-time
```

**Results**

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

# Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the DEFINE Token platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the DEFINE Token Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

# Closing Summary

In this report, we have considered the security of the DEFINE Token platform. We performed our audit according to the procedure described above.

The audit showed several high, medium, low, and informational severity issues. In the end, the majority of the issues were fixed and acknowledged by the Auditee. There are few remaining low and informational issues that require further discussion with the internal team; however, this issue has no bearing on the code's security.


QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com