Smart Contract Security Audit V1

RETHINK Token

1/2/2022

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN
_

Table of Contents

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Platform**: Avalanche's C-Chain
- **Contract Address**: 0xf1fC836B7345ACad53C9353861876Fa0A52952D0
- **Code Source:** https://snowtrace.io/address/0xf1fc836b7345acad53c9353861876fa0a52952d0#code

Token Information

- Name: $RETH
- Total Supply: 1,000,000,000,000,000
- Holders: address
- Total transactions:

Contracts address deployed to test net

RETHINK smart contract on test net

https://mumbai.polygonscan.com/address/0xe61056f6b1a50328efcb7058a4768ecf684b52b2

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**.because the team fix the critical issues in the contract

Well Secured	
Secured	✓
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 1 critical, 0 high, 0 medium, 3 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

RETHINK.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
RETHINK.sol	**a594028c189b29f2b14f191333193fac9da6486cc5390d697cc818daa700b153**	0xf1fC836B7345ACad53C9353861876Fa0A52952D0

- Contract: RETHINK
- Inherit: IBEP20, Auth
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
decimals	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
allowance	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
isOwner	✓	Read / public	**Passed**
pair	✓	Read / public	**Passed**
swapThreshold	✓	Read / public	**Passed**
router	✓	Read / public	**Passed**
_isFree	✓	Read / public	**Passed**
_maxTxAmount	✓	Read / public	**Passed**

marketingFeeReceiver	✓	Read / public	**Passed**
_maxWallet	✓	Read / public	**Passed**
launchedAt	✓	Read / public	**Passed**
launchedAtTimestamp	✓	Read / public	**Passed**
isOverLiquified	✓	Read / public	**Passed**
isAuthorized	✓	Read / public	**Passed**
totalFees	✓	Read / public	**Passed**
getTotalFee	✓	Read / public	**Passed**
getMultipliedFee	✓	Read / public	**Passed**
getLiquidityBacking	✓	Read / public	**Passed**
swapEnabled	✓	Read / public	**Passed**
MASK	✓	Read / public	**Passed**
autoBuybackEnabled	✓	Read / public	**Passed**
autoLiquidityReceiver	✓	Read / public	**Passed**
checkFree	✓	Read / public	**Passed**
distributorAddress	✓	Read / public	**Passed**
getCirculatingSupply	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
approveMax	✓	Write / public	**Passed**
authorize	✓	Write / public	**Passed**
clearBuybackMultiplier	✓	Write / public	**Passed**
launch	✓	Write / public	**Passed**
unSetFree	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setAutoBuybackSettings	✓	Write / public	**Passed**
unauthorize	✓	Write / public	**Passed**

triggerZeusBuyback	✓	Write / public	**Passed**
sweep	✓	Write / public	**Passed**
setTxLimit	✓	Write / public	**Passed**
setTargetLiquidity	✓	Write / public	**Passed**
setSwapBackSettings	✓	Write / public	**Passed**
setMaxWallet	✓	Write / public	**Passed**
setIsTxLimitExempt	✓	Write / public	**Passed**
setIsFeeExempt	✓	Write / public	**Passed**
setFees	✓	Write / public	**Passed**
setFree	✓	Write / public	**Passed**
setIsDividendExempt	✓	Write / public	**Passed**
setFeeReceivers	✓	Write / public	**Passed**
setBuybackMultiplierSettings	✓	Write / public	**Passed**
setDistributionCriteria	✓	Write / public	**Passed**
setDistributorSettings	✓	Write / public	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed with notes**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Front running.	**Passed**
6	Timestamp dependence.	**Passed with notes**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed with notes**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**Passed**
18	Design Logic.	**Passed**

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical:

#Reentrancy attack

Description

One of the major dangers of calling external contracts 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.

```
function transfer(address recipient, uint256 amount) external override returns
(bool) {

        return _transferFrom(msg.sender, recipient, amount);
    }
```

Status: closed. fixed in version 2.

High:

No High severity vulnerabilities were found

Medium:

No Medium severity vulnerabilities were found.

Low:

#Use of block.timestamp for comparisons

Description

The value of block.timestamp can be manipulated by the miner.
And conditions with strict equality is difficult to achieve - block.timestamp.

Remediation

Avoid use of block.timestamp

Status: Acknowledged

#Owner privileges (In the period when the owner isn't renounced)
Description

Owner can change Fees.
Owner can enable the trading.

```
function setTxLimit(uint256 amount) external authorized {
        require(amount >= _totalSupply / 1000);
        _maxTxAmount = amount;
    }

    function setIsDividendExempt(address holder, bool exempt) external authorized {
        require(holder != address(this) && holder != pair);
        isDividendExempt[holder] = exempt;
        if(exempt){
            distributor.setShare(holder, 0);
        }else{
            distributor.setShare(holder, _balances[holder]);
        }
    }

    function setIsFeeExempt(address holder, bool exempt) external authorized {
        isFeeExempt[holder] = exempt;
    }

    function setIsTxLimitExempt(address holder, bool exempt) external authorized {
        isTxLimitExempt[holder] = exempt;
    }

    function setFree(address holder) public onlyOwner {
        _isFree[holder] = true;
    }

    function unSetFree(address holder) public onlyOwner {
        _isFree[holder] = false;
    }

    function checkFree(address holder) public view onlyOwner returns(bool){
        return _isFree[holder];
    }

    function setFees(uint256 _liquidityFee, uint256 _buybackFee, uint256
_reflectionFee, uint256 _marketingFee, uint256 _feeDenominator) external authorized
{
        liquidityFee = _liquidityFee;
        buybackFee = _buybackFee;
        reflectionFee = _reflectionFee;
        marketingFee = _marketingFee;
        totalFee =
_liquidityFee.add(_buybackFee).add(_reflectionFee).add(_marketingFee);
        feeDenominator = _feeDenominator;
        require(totalFee < feeDenominator/4);
    }

    function setFeeReceivers(address _autoLiquidityReceiver, address
_marketingFeeReceiver) external authorized {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        marketingFeeReceiver = _marketingFeeReceiver;
    }

    function setSwapBackSettings(bool _enabled, uint256 _amount) external
authorized {
        swapEnabled = _enabled;
        swapThreshold = _amount;
    }

    function setTargetLiquidity(uint256 _target, uint256 _denominator) external
authorized {
        targetLiquidity = _target;
        targetLiquidityDenominator = _denominator;
    }

    function setDistributionCriteria(uint256 _minPeriod, uint256 _minDistribution)
external authorized {
        distributor.setDistributionCriteria(_minPeriod, _minDistribution);
    }

    function setDistributorSettings(uint256 gas) external authorized {
        require(gas < 750000);
        distributorGas = gas;
    }

    function getCirculatingSupply() public view returns (uint256) {
        return _totalSupply.sub(balanceOf(DEAD)).sub(balanceOf(ZERO));
    }
```

Remediation

Make these functions internal in next version or the team should announce the investors before change the fees and give them time if they want to use the old fees.

P.S: This issue is common to the majority of rewards smart contracts.

Status: Acknowledged.

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.0 instead of ^0.8.0). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

Remediation

Remove the ^ sign to lock the pragma version

Status: Closed. fixed in version2.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Naming Conventions

Description

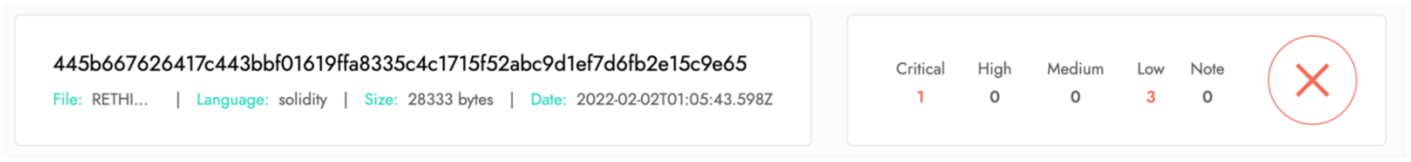
The contract follows a consistent naming convention where we are private variables with leading"_" and public variables without it. But we have missed to comply to the condition for certain variable names "__isFree" which is public

Remediation

Remove "_" from external variable names and add it to private variable names
Status: Acknowledged

Automatic Testing

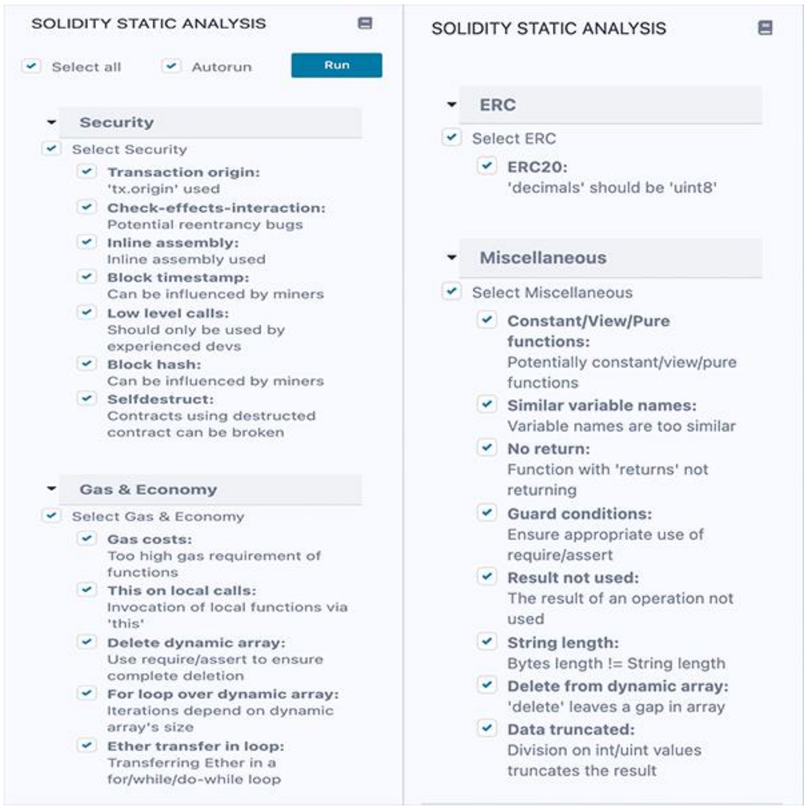
1- Check for security


445b667626417c443bbf01619ffa8335c4c1715f52abc9d1ef7d6fb2e15c9e65

File: RETHI... | Language: solidity | Size: 28333 bytes | Date: 2022-02-02T01:05:43.598Z

Critical	High	Medium	Low	Note
1	0	0	3	0

2- SOLIDITY STATIC ANALYSIS


SOLIDITY STATIC ANALYSIS

✓ Select all ✓ Autorun Run

Security

✓ Select Security

- ✓ **Transaction origin:** 'tx.origin' used
- ✓ **Check-effects-interaction:** Potential reentrancy bugs
- ✓ **Inline assembly:** Inline assembly used
- ✓ **Block timestamp:** Can be influenced by miners
- ✓ **Low level calls:** Should only be used by experienced devs
- ✓ **Block hash:** Can be influenced by miners
- ✓ **Selfdestruct:** Contracts using destructed contract can be broken

Gas & Economy

✓ Select Gas & Economy

- ✓ **Gas costs:** Too high gas requirement of functions
- ✓ **This on local calls:** Invocation of local functions via 'this'
- ✓ **Delete dynamic array:** Use require/assert to ensure complete deletion
- ✓ **For loop over dynamic array:** Iterations depend on dynamic array's size
- ✓ **Ether transfer in loop:** Transferring Ether in a for/while/do-while loop

SOLIDITY STATIC ANALYSIS

ERC

✓ Select ERC

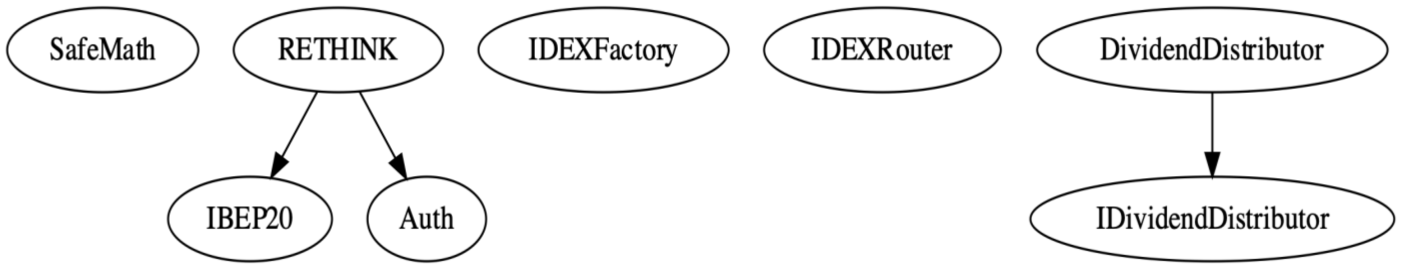
- ✓ **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

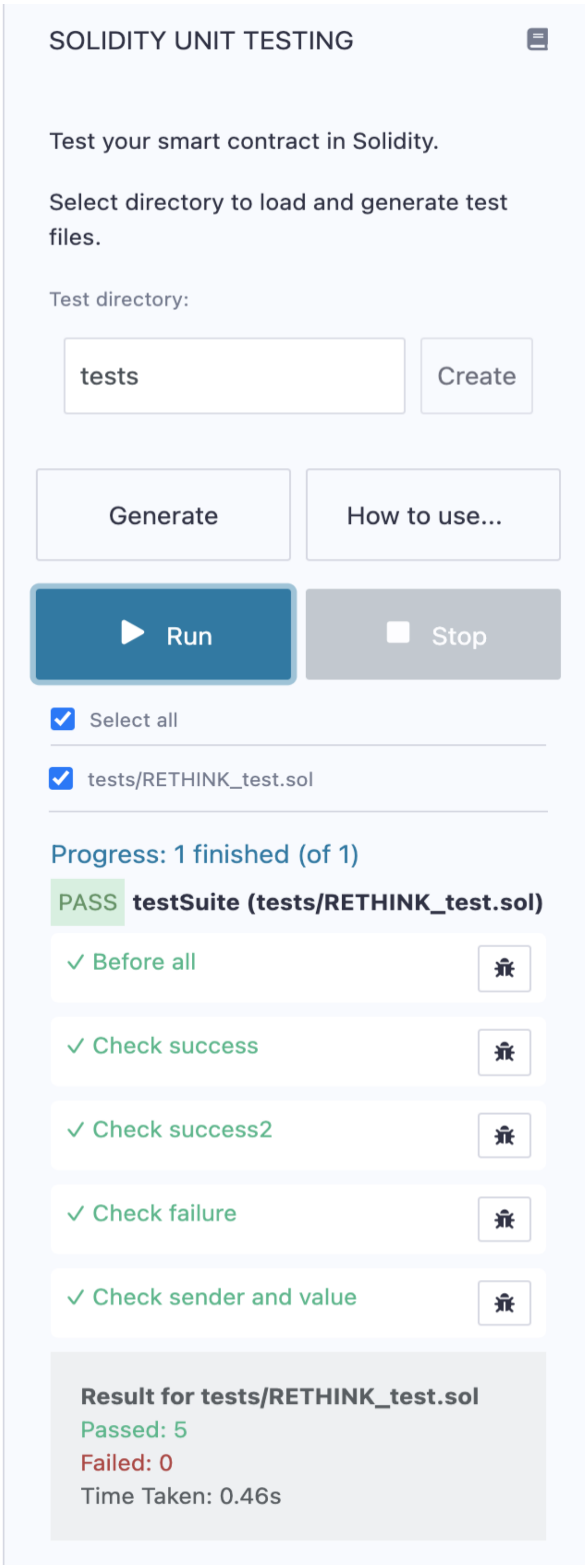
✓ Select Miscellaneous

- ✓ **Constant/View/Pure functions:** Potentially constant/view/pure functions
- ✓ **Similar variable names:** Variable names are too similar
- ✓ **No return:** Function with 'returns' not returning
- ✓ **Guard conditions:** Ensure appropriate use of require/assert
- ✓ **Result not used:** The result of an operation not used
- ✓ **String length:** Bytes length != String length
- ✓ **Delete from dynamic array:** 'delete' leaves a gap in array
- ✓ **Data truncated:** Division on int/uint values truncates the result

3- Inheritance graph



4- SOLIDITY UNIT TESTING


SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

✓ Select all

✓ tests/RETHINK_test.sol

Progress: 1 finished (of 1)

PASS **testSuite (tests/RETHINK_test.sol)**

✓ Before all

✓ Check success

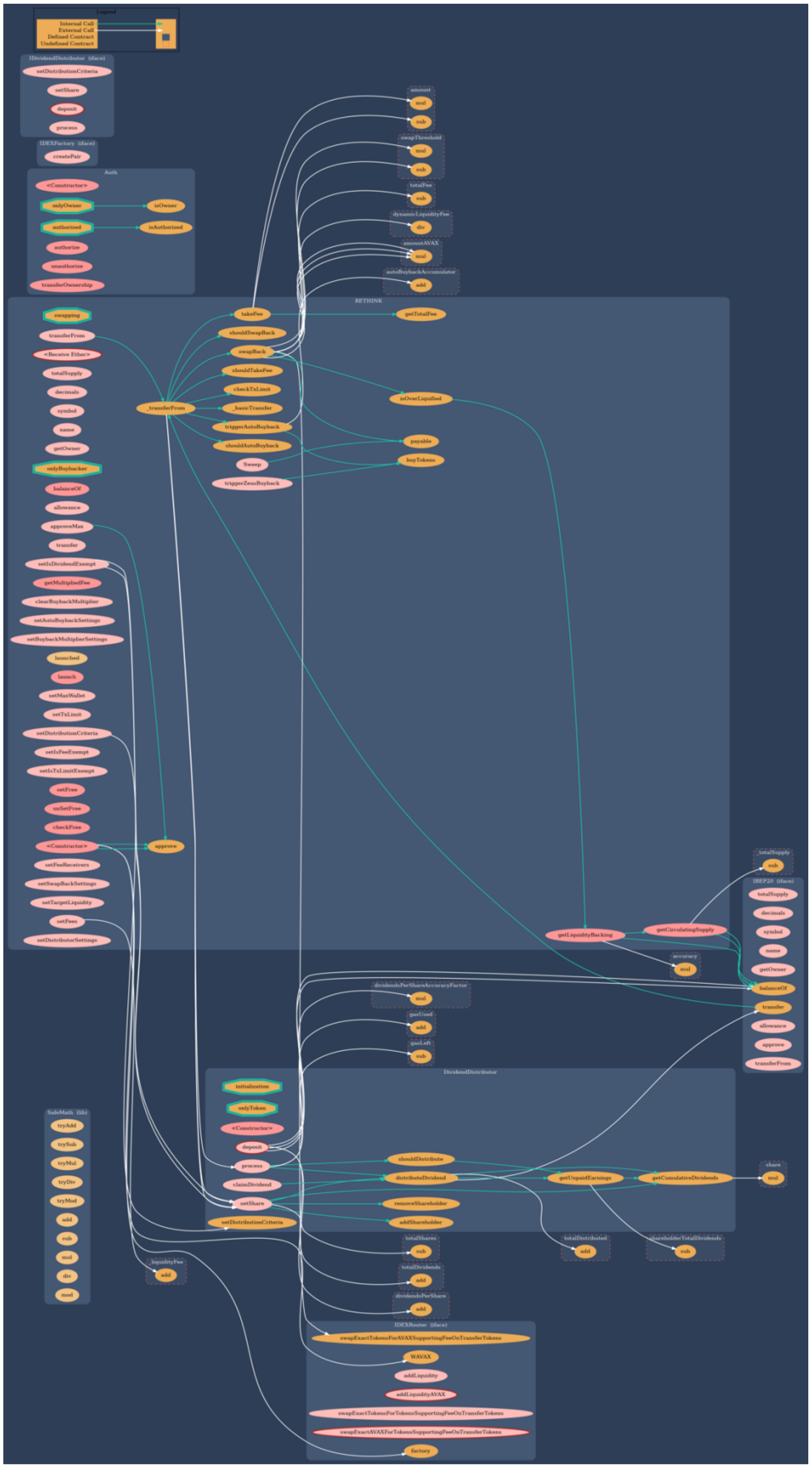
✓ Check success2

✓ Check failure

✓ Check sender and value

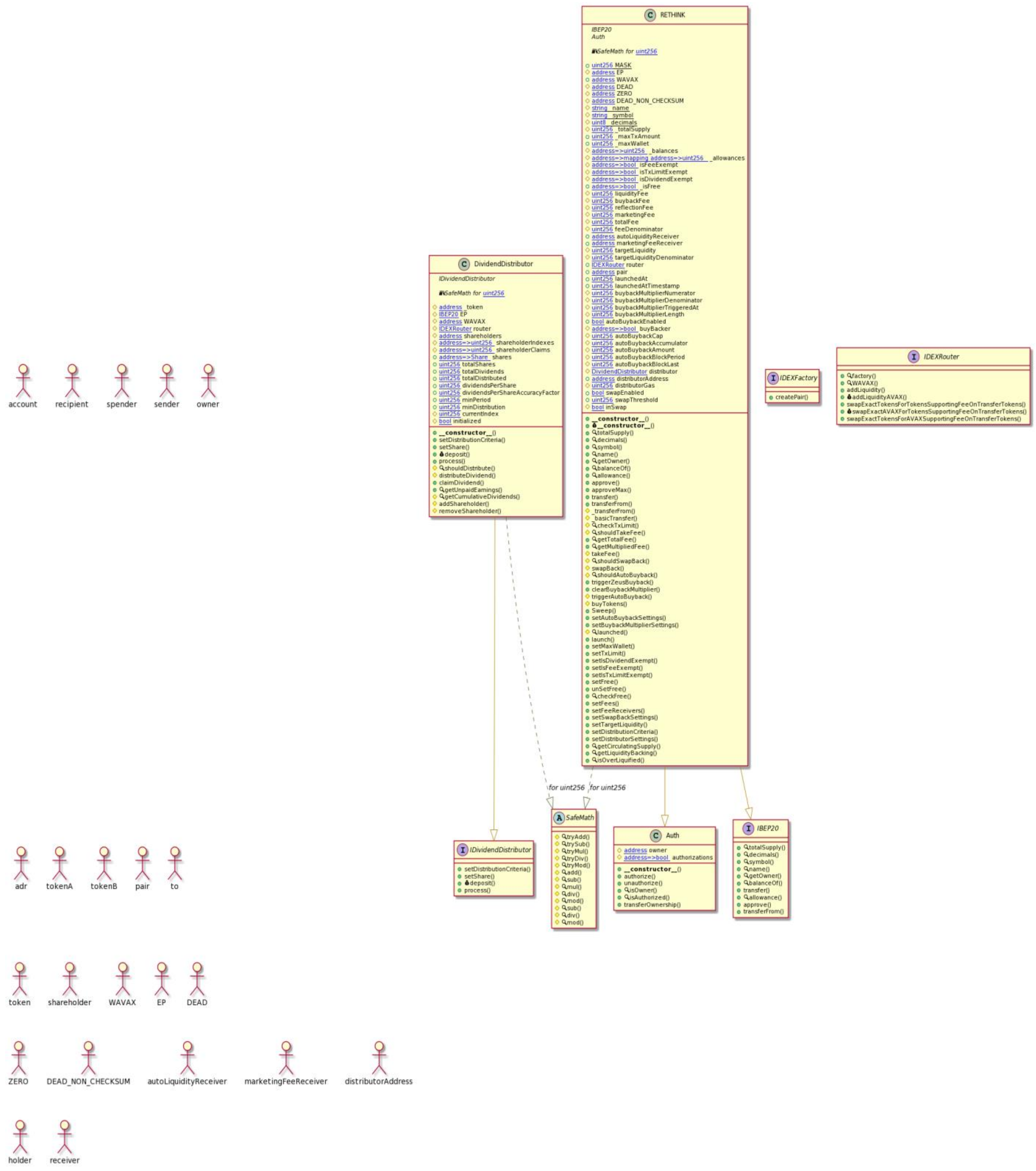
Result for tests/RETHINK_test.sol
Passed: 5
Failed: 0
Time Taken: 0.46s

5- Call graph



Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IDividendDistributor (iface)
setDistributionCriteria
setShare
deposit
process
IDEXFactory (iface)
createPair
Auth
<Constructor>
onlyOwner
isOwner
authorized
isAuthorized
authorize
unauthorize
transferOwnership
amount
mul
sub
swapThreshold
totalFee
dynamicLiquidityFee
div
amountAVAX
autoBuybackAccumulator
add
RETHINK
swapping
takeFee
getTotalFee
transferFrom
shouldSwapBack
<Receive Ether>
swapBack
totalSupply
shouldTakeFee
decimals
checkTxLimit
isOverLiquified
symbol
_transferFrom
_basicTransfer
name
triggerAutoBuyback
getOwner
shouldAutoBuyback
payable
Sweep
buyTokens
onlyBuybacker
triggerZeusBuyback
balanceOf
allowance
approveMax
transfer
setIsDividendExempt
getMultipliedFee
clearBuybackMultiplier
setAutoBuybackSettings
setBuybackMultiplierSettings
launched
launch
setMaxWallet
setTxLimit
setDistributionCriteria
setIsFeeExempt
setIsTxLimitExempt
setFree
unSetFree
checkFree
<Constructor>
approve
setFeeReceivers
setSwapBackSettings
setTargetLiquidity
setFees
setDistributorSettings
_totalSupply
IBEP20 (iface)
getLiquidityBacking
getCirculatingSupply
accuracy
dividendsPerShareAccuracyFactor
gasUsed
gasLeft
DividendDistributor
initialization
onlyToken
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
mod
claimDividend
shouldDistribute
distributeDividend
getUnpaidEarnings
getCumulativeDividends
share
removeShareholder
addShareholder
totalShares
totalDistributed
shareholderTotalDividends
liquidityFee
totalDividends
dividendsPerShare
IDEXRouter (iface)
swapExactTokensForAVAXSupportingFeeOnTransferTokens
WAVAX
addLiquidity
addLiquidityAVAX
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactAVAXForTokensSupportingFeeOnTransferTokens
factory

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
884557bf  =>  tryAdd(uint256,uint256)
a29962b1  =>  trySub(uint256,uint256)
6281efa4  =>  tryMul(uint256,uint256)
736ecb18  =>  tryDiv(uint256,uint256)
38dc0867  =>  tryMod(uint256,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
f43f523a  =>  mod(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
b745d336  =>  div(uint256,uint256,string)
71af23e8  =>  mod(uint256,uint256,string)
18160ddd  =>  totalSupply()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
893d20e8  =>  getOwner()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
b6a5d7de  =>  authorize(address)
f0b37c04  =>  unauthorize(address)
2f54bf6e  =>  isOwner(address)
fe9fbb80  =>  isAuthorized(address)
f2fde38b  =>  transferOwnership(address)
c9c65396  =>  createPair(address,address)
c45a0155  =>  factory()
73b295c2  =>  WAVAX()
e8e33700  =>
addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f91b3f72  =>  addLiquidityAVAX(address,uint256,uint256,uint256,address,uint256)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],add
ress,uint256)
c57559dd  =>
swapExactAVAXForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint2
56)
762b1562  =>
swapExactTokensForAVAXSupportingFeeOnTransferTokens(uint256,uint256,address[],addre
ss,uint256)
2d48e896  =>  setDistributionCriteria(uint256,uint256)
14b6ca96  =>  setShare(address,uint256)
d0e30db0  =>  deposit()
ffb2c479  =>  process(uint256)
8c21cd52  =>  shouldDistribute(address)
5319504a  =>  distributeDividend(address)
f0fc6bca  =>  claimDividend()
28fd3198  =>  getUnpaidEarnings(address)
e68af3ac  =>  getCumulativeDividends(uint256)
db29fe12  =>  addShareholder(address)
9babdad6  =>  removeShareholder(address)
```

```
571ac8b0  =>  approveMax(address)
cb712535  =>  _transferFrom(address,address,uint256)
f0774e71  =>  _basicTransfer(address,address,uint256)
4afa518a  =>  checkTxLimit(address,uint256)
e7c44c69  =>  shouldTakeFee(address)
f1f3bca3  =>  getTotalFee(bool)
d806d12f  =>  getMultipliedFee()
20cb7bce  =>  takeFee(address,address,uint256)
0d5c6cea  =>  shouldSwapBack()
6ac5eeee  =>  swapBack()
4d4e6fe5  =>  shouldAutoBuyback()
f5cfec0a  =>  triggerZeusBuyback(uint256,bool)
b210b06d  =>  clearBuybackMultiplier()
5cd44665  =>  triggerAutoBuyback()
c625e9b1  =>  buyTokens(uint256,address)
7088fb7f  =>  Sweep()
048c7baf  =>  setAutoBuybackSettings(bool,uint256,uint256,uint256)
2375ce40  =>  setBuybackMultiplierSettings(uint256,uint256,uint256)
8091f3bf  =>  launched()
01339c21  =>  launch()
5d0044ca  =>  setMaxWallet(uint256)
5c85974f  =>  setTxLimit(uint256)
f708a64f  =>  setIsDividendExempt(address,bool)
658d4b7f  =>  setIsFeeExempt(address,bool)
f84ba65d  =>  setIsTxLimitExempt(address,bool)
89ef69f6  =>  setFree(address)
3425c001  =>  unSetFree(address)
07c89fe8  =>  checkFree(address)
04a66b48  =>  setFees(uint256,uint256,uint256,uint256,uint256)
a4b45c00  =>  setFeeReceivers(address,address)
df20fd49  =>  setSwapBackSettings(bool,uint256)
201e7991  =>  setTargetLiquidity(uint256,uint256)
9d1944f5  =>  setDistributorSettings(uint256)
2b112e49  =>  getCirculatingSupply()
d51ed1c8  =>  getLiquidityBacking(uint256)
1161ae39  =>  isOverLiquified(uint256,uint256)
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/RETHINK.sol |
c8382c4f66fd2be50ac39633468770950bc0e36e |
| /Users/macbook/Desktop/smart contracts/presale .sol |
fa1ea7724f65a2ef484730cccb28a880c8853b5d |

 Contracts Description Table

|  Contract  |         Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
--------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **SafeMath** | Library |  |||
| └ | tryAdd | Internal 🔒 |   | |
| └ | trySub | Internal 🔒 |   | |
| └ | tryMul | Internal 🔒 |   | |
| └ | tryDiv | Internal 🔒 |   | |
| └ | tryMod | Internal 🔒 |   | |
| └ | add | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | mul | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
||||||
| **IBEP20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | name | External ❗ |   |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **Auth** | Implementation |  |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | authorize | Public ❗ | 🛑  | onlyOwner |
| └ | unauthorize | Public ❗ | 🛑  | onlyOwner |
| └ | isOwner | Public ❗ |   |NO❗ |
| └ | isAuthorized | Public ❗ |   |NO❗ |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
||||||
| **IDEXFactory** | Interface |  |||
| └ | createPair | External ❗ | 🛑  |NO❗ |
||||||
| **IDEXRouter** | Interface |  |||
| └ | factory | External ❗️ |   |NO❗️ |
| └ | WAVAX | External ❗️ |   |NO❗️ |
| └ | addLiquidity | External ❗️ | 🛑  |NO❗️ |
| └ | addLiquidityAVAX | External ❗️ |  💵 |NO❗️ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗️ | 🛑  |NO❗️
|
| └ | swapExactAVAXForTokensSupportingFeeOnTransferTokens | External ❗️ |  💵 |NO❗️ |
| └ | swapExactTokensForAVAXSupportingFeeOnTransferTokens | External ❗️ | 🛑  |NO❗️ |
||||||
| **IDividendDistributor** | Interface |  |||
| └ | setDistributionCriteria | External ❗️ | 🛑  |NO❗️ |
| └ | setShare | External ❗️ | 🛑  |NO❗️ |
| └ | deposit | External ❗️ |  💵 |NO❗️ |
| └ | process | External ❗️ | 🛑  |NO❗️ |
||||||
| **DividendDistributor** | Implementation | IDividendDistributor |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | setDistributionCriteria | External ❗️ | 🛑  | onlyToken |
| └ | setShare | External ❗️ | 🛑  | onlyToken |
| └ | deposit | External ❗️ |  💵 | onlyToken |
| └ | process | External ❗️ | 🛑  | onlyToken |
| └ | shouldDistribute | Internal 🔒 |   | |
| └ | distributeDividend | Internal 🔒 | 🛑  | |
| └ | claimDividend | External ❗️ | 🛑  |NO❗️ |
| └ | getUnpaidEarnings | Public ❗️ |   |NO❗️ |
| └ | getCumulativeDividends | Internal 🔒 |   | |
| └ | addShareholder | Internal 🔒 | 🛑  | |
| └ | removeShareholder | Internal 🔒 | 🛑  | |
||||||
| **RETHINK** | Implementation | IBEP20, Auth |||
| └ | <Constructor> | Public ❗️ | 🛑  | Auth |
| └ | <Receive Ether> | External ❗️ |  💵 |NO❗️ |
| └ | totalSupply | External ❗️ |   |NO❗️ |
| └ | decimals | External ❗️ |   |NO❗️ |
| └ | symbol | External ❗️ |   |NO❗️ |
| └ | name | External ❗️ |   |NO❗️ |
| └ | getOwner | External ❗️ |   |NO❗️ |
| └ | balanceOf | Public ❗️ |   |NO❗️ |
| └ | allowance | External ❗️ |   |NO❗️ |
| └ | approve | Public ❗️ | 🛑  |NO❗️ |
| └ | approveMax | External ❗️ | 🛑  |NO❗️ |
| └ | transfer | External ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | External ❗️ | 🛑  |NO❗️ |
| └ | _transferFrom | Internal 🔒 | 🛑  | |
| └ | _basicTransfer | Internal 🔒 | 🛑  | |
| └ | checkTxLimit | Internal 🔒 |   | |
| └ | shouldTakeFee | Internal 🔒 |   | |
| └ | getTotalFee | Public ❗️ |   |NO❗️ |
| └ | getMultipliedFee | Public ❗️ |   |NO❗️ |
| └ | takeFee | Internal 🔒 | 🛑  | |
| └ | shouldSwapBack | Internal 🔒 |   | |
| └ | swapBack | Internal 🔒 | 🛑  | swapping |
| └ | shouldAutoBuyback | Internal 🔒 |   | |
| └ | triggerZeusBuyback | External ❗️ | 🛑  | authorized |
| └ | clearBuybackMultiplier | External ❗️ | 🛑  | authorized |
| └ | triggerAutoBuyback | Internal 🔒 | 🛑  | |
```

└	buyTokens	Internal 🔒	🛑	swapping
└	Sweep	External ❗️	🛑	onlyOwner
└	setAutoBuybackSettings	External ❗️	🛑	authorized
└	setBuybackMultiplierSettings	External ❗️	🛑	authorized
└	launched	Internal 🔒		
└	launch	Public ❗️	🛑	authorized
└	setMaxWallet	External ❗️	🛑	authorized
└	setTxLimit	External ❗️	🛑	authorized
└	setIsDividendExempt	External ❗️	🛑	authorized
└	setIsFeeExempt	External ❗️	🛑	authorized
└	setIsTxLimitExempt	External ❗️	🛑	authorized
└	setFree	Public ❗️	🛑	onlyOwner
└	unSetFree	Public ❗️	🛑	onlyOwner
└	checkFree	Public ❗️		onlyOwner
└	setFees	External ❗️	🛑	authorized
└	setFeeReceivers	External ❗️	🛑	authorized
└	setSwapBackSettings	External ❗️	🛑	authorized
└	setTargetLiquidity	External ❗️	🛑	authorized
└	setDistributionCriteria	External ❗️	🛑	authorized
└	setDistributorSettings	External ❗️	🛑	authorized
└	getCirculatingSupply	Public ❗️		NO❗️
└	getLiquidityBacking	Public ❗️		NO❗️
└	isOverLiquified	Public ❗️		NO❗️

||||||

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “secured”.

✓ No mint function.
✓ No volatile code.
✓ Not many high severity issues were found.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn’t say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.