SMART CONTRACT AUDIT REPORT

For

Wyss Coin (WYS)

Prepared By: SaferICO Team

Prepared on: 11/1/2022

Prepared for: Wyss Coin Team

Contract address:
0xe7D0e0ED7E53aed243CB80921AE1cE3be5661e2C

Table of Content

- ## 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) 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.

• Scope of the audit

The scope of this audit was to analyze and document the **Wyss Coin (WYS)** smart contract codebase for quality, security, and correctness.

• Introduction

During the period of **January 10, 2022, to January 11, 2022** - SaferICO Team performed a security audit for **Wyss Coin (WYS)** smart contracts.

The project has 1 file. It contains approx 2206 lines of Solidity code. Most of the functions and state variables are well commented on using the Nat spec documentation, but that does not create any vulnerability.

Source code: https://github.com/WyssCoin/WyssCoin/blob/main/wysscoin%20with%20blacklist%20function%20(1).txt

Check Vulnerabilities

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices automatically.

1. Unit tests passing.

2. Compilator warnings;

3. Race Conditions. Reentrancy. Cross-function Race Conditions. Pitfalls in Race Condition solutions;

4. Possible delays in data delivery;

5. Transaction-Ordering Dependence (front running);

6. Timestamp Dependence;

7. Integer Overflow and Underflow;

8. DoS with (unexpected) Revert;

9. DoS with Block Gas Limit

10. Call Depth Attack. Not relevant in modern ethereum network

11. Methods execution permissions;

12. Oracles calls;

13. Economy model. It’s important to forecast scenarios when a user is provided with additional economic motivation or faced with limitations. If application logic is based on incorrect economy model, the application will not function correctly and participants will incur financial losses. This type of issue is most often found in bonus rewards systems.

14. The impact of the exchange rate on the logic;

15. Private user data leaks.

• Issue Categories

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

Risk-level	Description
High	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 be fixed before moving to a live environment.
Medium	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	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 issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

- # Issues Found – Code Review

High severity issues

There are no High severity vulnerabilities found

Medium severity issues

There are no Medium severity vulnerabilities found

Low severity issues

#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

#Contract has many pragmas and the main Pragma version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.10 instead of ^0.8.10). 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. and the contract has many libraries with many pragmas.

Remediation

Remove the ^ sign to lock the pragma version and make all libraries with same pragma.

Status: Closed. Because the developer deployed the contract with latest pragma which more secure and Uniswap libraries doesn't have new code with 8 pragma or above so he had to use the old library which it is secure to use.

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

Description

Owner can Excludes or includes an address from receiving reflections.

Owner can enable the trading.

Owner can Exclude or include an address from fees, etc.

Owner can change all fees.

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
        _isExcludedFromFees[account] = excluded;


        emit ExcludeFromFees(account, excluded);}
    function excludeMultipleAccountsFromFees(
        address[] calldata accounts,
        bool excluded
    ) public onlyOwner {
        for (uint256 i = 0; i < accounts.length; i++) {
            _isExcludedFromFees[accounts[i]] = excluded; }
        emit ExcludeMultipleAccountsFromFees(accounts, excluded);}
function setFees(
        uint8 _market,
        uint8 _liq,
        uint8 _reward,
        uint8 _buyback
    ) external onlyOwner {
        marketingFee = _market;
        liquidityFee = _liq;
        rewardsFee = _reward;
        buybackFee = _buyback;}
function enableTrading() external onlyOwner {
        isTradingEnabled = true; }
```

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.

Status: Acknowledged

Informational issues

#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 like "__isBlacklisted" which is public

Remediation

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

Status: Acknowledged

Constant calculations in the contract
Description
recalculated initialization will save 2847 units of gas in deployment

```
uint256 public swapTokensAtAmount = 2 * 10**5 * (10**9);
    uint256 public maxBuyAmount = 1 * 10**6 * 10**9;
    uint256 public maxSellAmount = 1 * 10**6 * 10**9;
    uint256 public maxWalletAmount = 5 * 10**6 * 10**9;
```

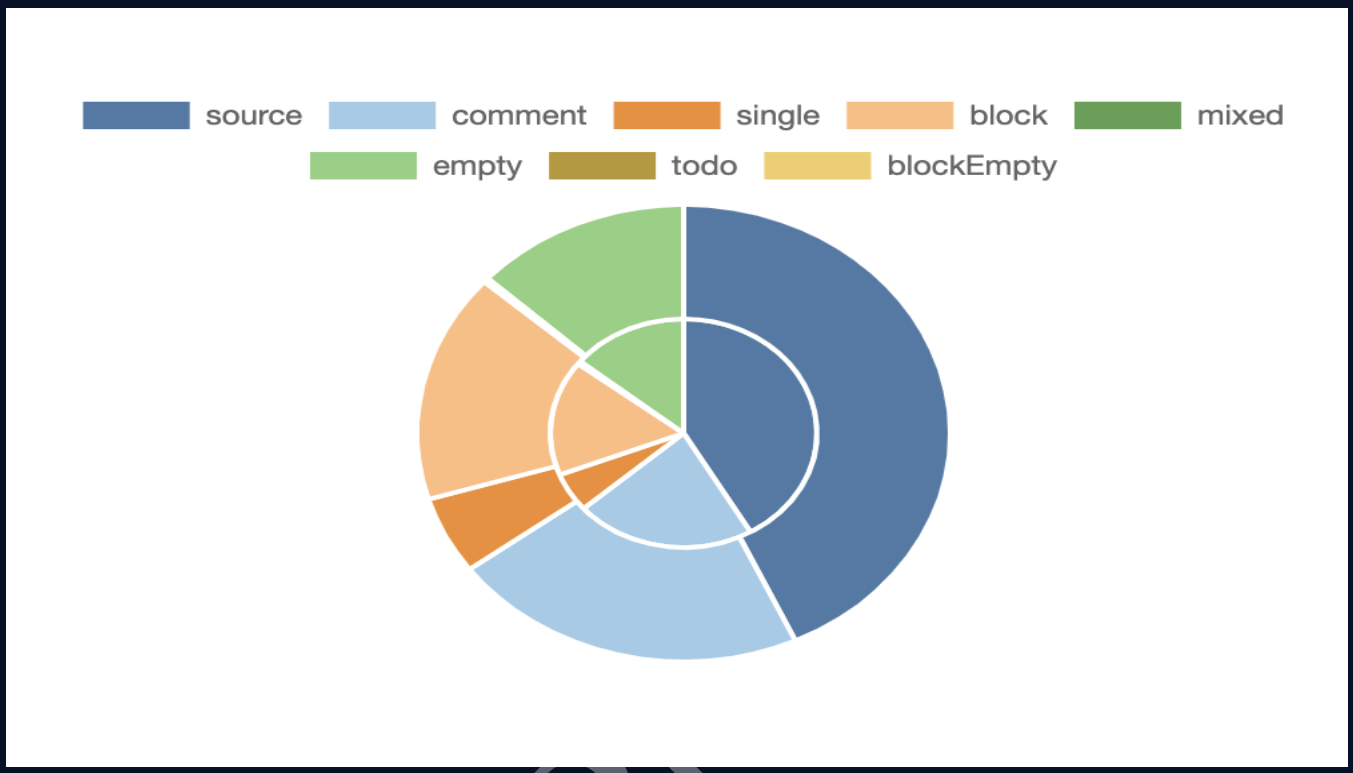
Recommendation

Replace the initialization as

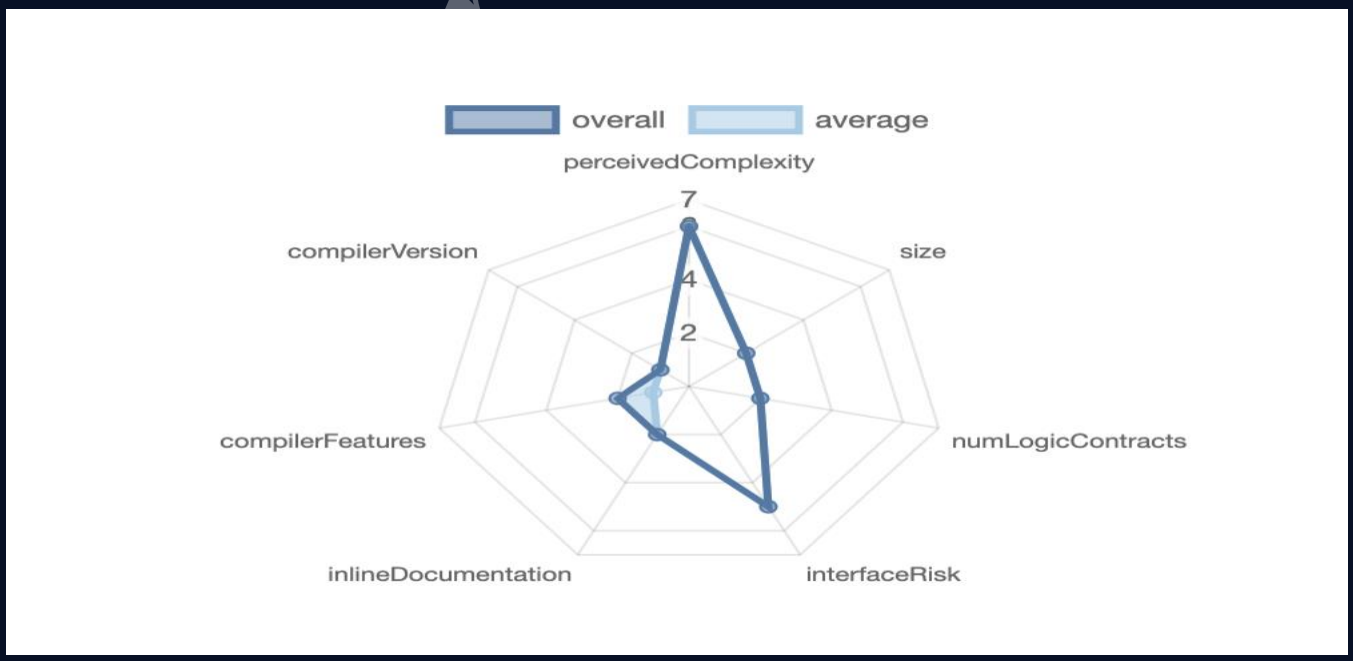
```
uint256 public swapTokensAtAmount = 200000000000000;
    uint256 public maxBuyAmount = 1000000000000000;
    uint256 public maxSellAmount = 1000000000000000;
    uint256 public maxWalletAmount = 5000000000000000;
```

Status: Acknowledged

- **Source Lines**

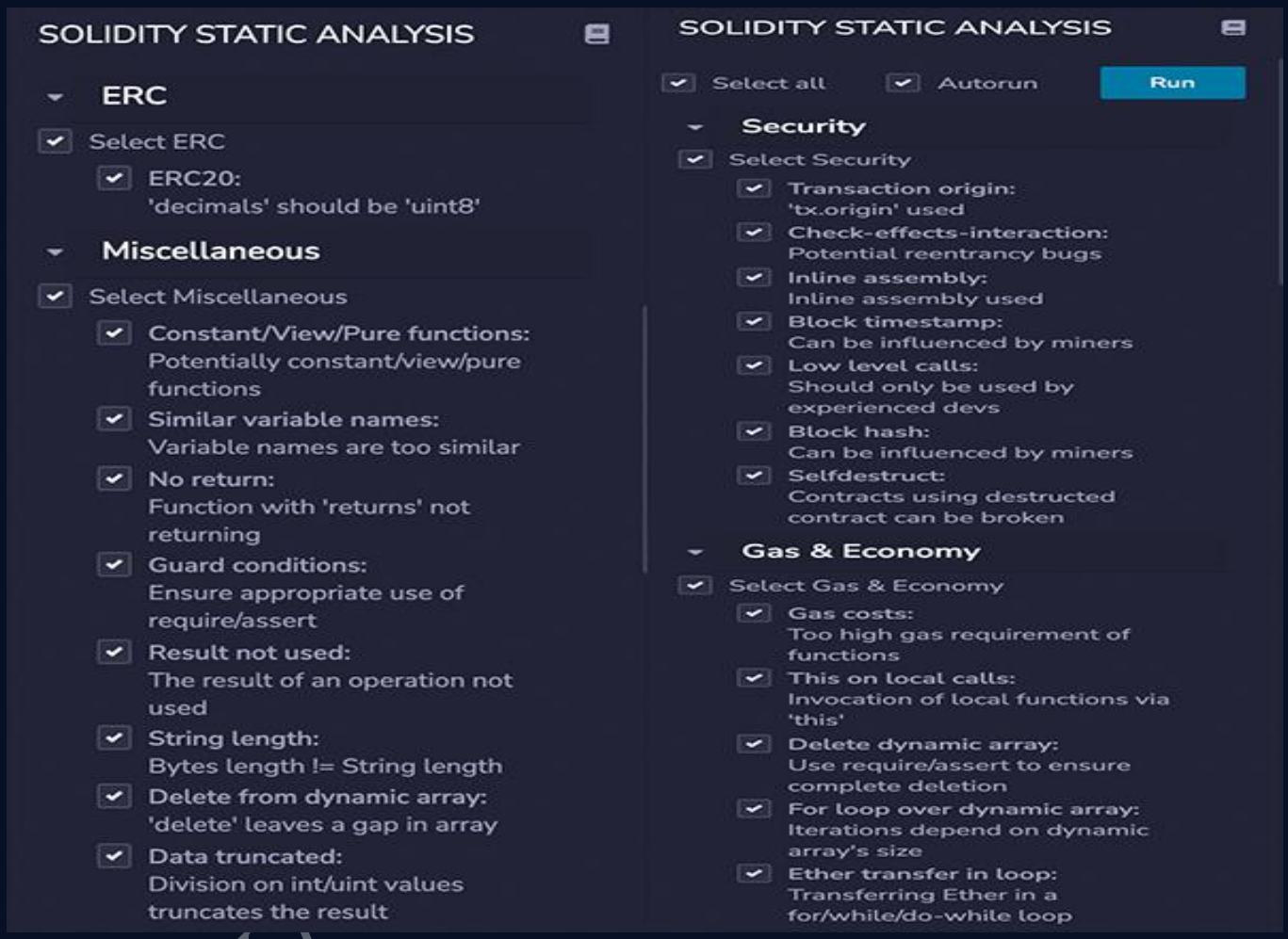


- **Risk Level**

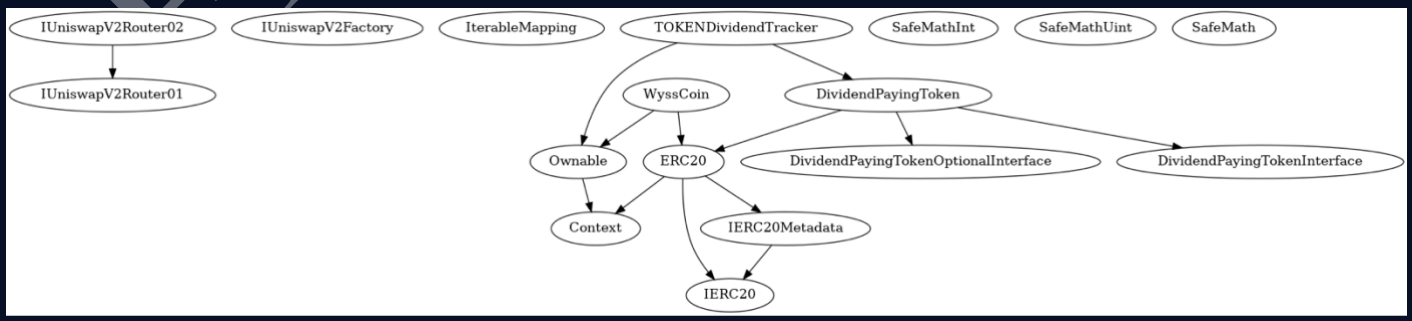


- **Testing proves**

- **Solidity Static Analysis**

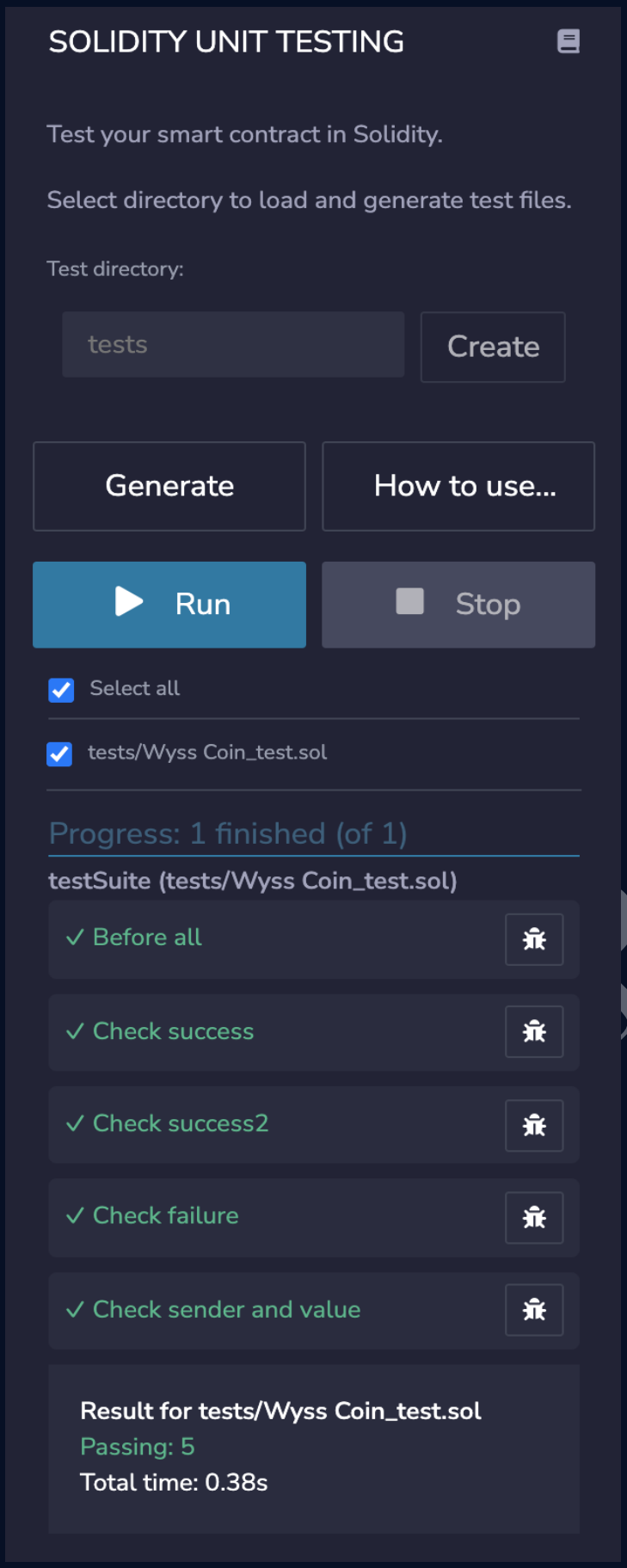


- **Inheritance**



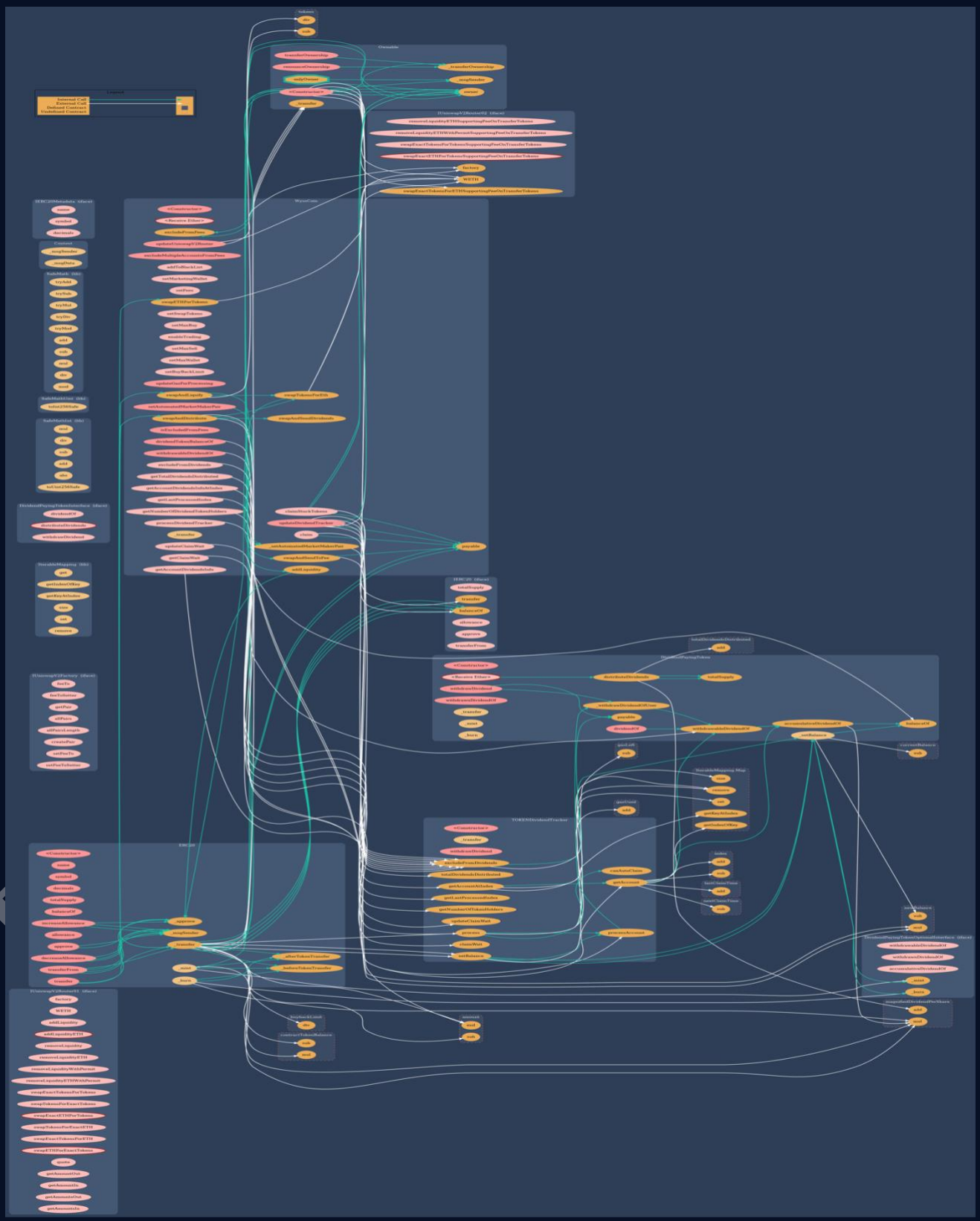
- **Solidity Unit Testing Code & Results**

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.4.22 <0.9.0;

// This import is automatically injected by Remix
import "remix_tests.sol";

// This import is required to use custom transaction context
// Although it may fail compilation in 'Solidity Compiler' plugin
// But it will work fine in 'Solidity Unit Testing' plugin
import "remix_accounts.sol";
import "../Wyss Coin.sol";

// File name has to end with '_test.sol', this file can contain more than one testSuite contracts
contract testSuite {

    /// 'beforeAll' runs before all other tests
    /// More special functions are: 'beforeEach', 'beforeAll', 'afterEach' & 'afterAll'
    function beforeAll() public {
        // <instantiate contract>
        Assert.equal(uint(1), uint(1), "1 should be equal to 1");
    }

    function checkSuccess() public {
        // Use 'Assert' methods: https://remix-ide.readthedocs.io/en/latest/assert_library.html
        Assert.ok(2 == 2, 'should be true');
        Assert.greaterThan(uint(2), uint(1), "2 should be greater than to 1");
        Assert.lesserThan(uint(2), uint(3), "2 should be lesser than to 3");
    }

    function checkSuccess2() public pure returns (bool) {
        // Use the return value (true or false) to test the contract
        return true;
    }
```

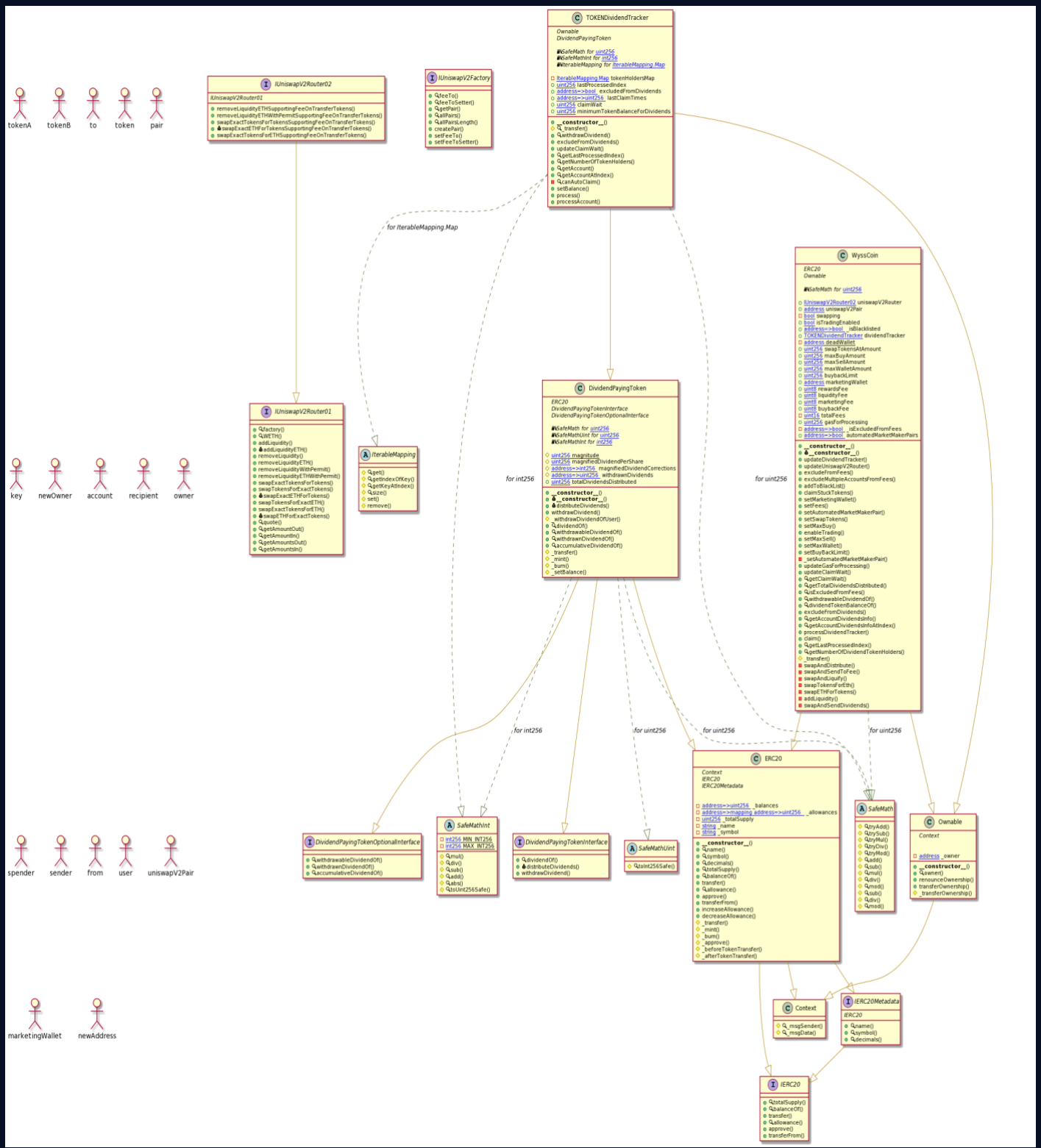



```
    function checkFailure() public {
        Assert.notEqual(uint(1), uint(2), "1 should not be equal to 1");
    }

    /// Custom Transaction Context: https://remix-ide.readthedocs.io/en/latest/unittesting.html#customization
    /// #sender: account-1
    /// #value: 100
    function checkSenderAndValue() public payable {
        // account index varies 0-9, value is in wei
        Assert.equal(msg.sender, TestsAccounts.getAccount(1), "Invalid sender");
        Assert.equal(msg.value, 100, "Invalid value");
    }
}
```

- **Call Graph**



- **Unified Modeling Language (UML)**



- ## Capabilities

Components

Contracts	Libraries	Interfaces	Abstract
4	4	7	2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Public	Payable
108	8

External	Internal	Private	Pure	View
74	123	9	27	48

StateVariables

Total	Public
40	23

Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`>=0.6.2` `>=0.5.0` `^0.8.8` `^0.8.10` `^0.8.0`		`yes`		

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
`yes`					`yes` → `NewContract:TOKENDividendTracker`

TryCatch	Unchecked
`yes`	`yes`

- ## Source Units In Scope

Source Units in Scope

Source Units Analyzed: `1`
Source Units in Scope: `1` **(100%)**

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	Wyss Coin.sol	10	7	2206	1775	957	617	752	
	Totals	**10**	**7**	**2206**	**1775**	**957**	**617**	**752**	

Legend: [−]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

- **Function Signature**

```
39509351 => increaseAllowance(address,uint256)
43509138 => div(int256,int256)
c45a0155 => factory()
ad5c4648 => WETH()
e8e33700 => addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719 => addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde => removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec => removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c => removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
ded9382a => removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
38ed1739 => swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee => swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5 => swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a => swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5 => swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41 => swapETHForExactTokens(uint256,address[],address,uint256)
ad615dec => quote(uint256,uint256,uint256)
054d50d4 => getAmountOut(uint256,uint256,uint256)
85f8c259 => getAmountIn(uint256,uint256,uint256)
d06ca61f => getAmountsOut(uint256,address[])
1f00ca74 => getAmountsIn(uint256,address[])
af2979eb => removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
5b0d5984 =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes
32,bytes32)
5c11d795 => swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95 => swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947 => swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
017e7e58 => feeTo()
094b7415 => feeToSetter()
e6a43905 => getPair(address,address)
1e3dd18b => allPairs(uint256)
574f2ba3 => allPairsLength()
c9c65396 => createPair(address,address)
f46901ed => setFeeTo(address)
```

a2e74af6 => setFeeToSetter(address)

268d8e2e => get(Map,address)

b45dad3d => getIndexOfKey(Map,address)

7596720f => getKeyAtIndex(Map,uint256)

b1b533f3 => size(Map)

6b06f325 => set(Map,address,uint256)

0eac8729 => remove(Map,address)

a8b9d240 => withdrawableDividendOf(address)

aafd847a => withdrawnDividendOf(address)

27ce0147 => accumulativeDividendOf(address)

91b89fba => dividendOf(address)

03c83302 => distributeDividends()

6a474002 => withdrawDividend()

bbe93d91 => mul(int256,int256)

adefc37b => sub(int256,int256)

a5f3c23b => add(int256,int256)

1b5ac4b5 => abs(int256)

744f7c7d => toUint256Safe(int256)

e823b9bf => toInt256Safe(uint256)

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)

119df25f => _msgSender()

8b49d47e => _msgData()

8da5cb5b => owner()

715018a6 => renounceOwnership()

f2fde38b => transferOwnership(address)

d29d44ee => _transferOwnership(address)

```
18160ddd => totalSupply()
70a08231 => balanceOf(address)
a9059cbb => transfer(address,uint256)
dd62ed3e => allowance(address,address)
095ea7b3 => approve(address,uint256)
23b872dd => transferFrom(address,address,uint256)
06fdde03 => name()
95d89b41 => symbol()
313ce567 => decimals()
a457c2d7 => decreaseAllowance(address,uint256)
30e0789e => _transfer(address,address,uint256)
4e6ec247 => _mint(address,uint256)
6161eb18 => _burn(address,uint256)
104e81ff => _approve(address,address,uint256)
cad3be83 => _beforeTokenTransfer(address,address,uint256)
8f811a1c => _afterTokenTransfer(address,address,uint256)
373de4aa => _withdrawDividendOfUser(address)
ab86e0a6 => _setBalance(address,uint256)
88bdd9be => updateDividendTracker(address)
65b8dbc0 => updateUniswapV2Router(address)
c0246668 => excludeFromFees(address,bool)
c492f046 => excludeMultipleAccountsFromFees(address[],bool)
417c73a7 => addToBlackList(address)
f9d0831a => claimStuckTokens(address)
5d098b38 => setMarketingWallet(address)
9faa7cfd => setFees(uint8,uint8,uint8,uint8)
9a7a23d6 => setAutomatedMarketMakerPair(address,bool)
bf6642e7 => setSwapTokens(uint256)
f53bc835 => setMaxBuy(uint256)
8a8c523c => enableTrading()
ef998cf0 => setMaxSell(uint256)
5d0044ca => setMaxWallet(uint256)
e7deff7f => setBuyBackLimit(uint256)
a7f7b36f => _setAutomatedMarketMakerPair(address,bool)
871c128d => updateGasForProcessing(uint256)
e98030c7 => updateClaimWait(uint256)
a26579ad => getClaimWait()
30bb4cff => getTotalDividendsDistributed()
```

```
4fbee193 => isExcludedFromFees(address)
6843cd84 => dividendTokenBalanceOf(address)
31e79db0 => excludeFromDividends(address)
ad56c13c => getAccountDividendsInfo(address)
f27fd254 => getAccountDividendsInfoAtIndex(uint256)
700bb191 => processDividendTracker(uint256)
4e71d92d => claim()
e7841ec0 => getLastProcessedIndex()
64b0f653 => getNumberOfDividendTokenHolders()
8b5d5cab => swapAndDistribute(uint256)
173684b4 => swapAndSendToFee(uint256,uint16)
173865ad => swapAndLiquify(uint256)
b28805f4 => swapTokensForEth(uint256)
2eab2841 => swapETHForTokens(uint256)
9cd441da => addLiquidity(uint256,uint256)
818c19dc => swapAndSendDividends(uint256)
09bbedde => getNumberOfTokenHolders()
fbcbc0f1 => getAccount(address)
5183d6fd => getAccountAtIndex(uint256)
77fdb837 => canAutoClaim(uint256)
e30443bc => setBalance(address,uint256)
ffb2c479 => process(uint256)
bc4c4b37 => processAccount(address,bool)
```

- # Automatic General Report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Wyss Coin.sol	ded4602550b49fcc572e90c54f32dbf747bd8611

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IUniswapV2Router01	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🔴	NO❗
└	addLiquidityETH	External ❗	💷	NO❗
└	removeLiquidity	External ❗	🔴	NO❗
└	removeLiquidityETH	External ❗	🔴	NO❗
└	removeLiquidityWithPermit	External ❗	🔴	NO❗
└	removeLiquidityETHWithPermit	External ❗	🔴	NO❗
└	swapExactTokensForTokens	External ❗	🔴	NO❗
└	swapTokensForExactTokens	External ❗	🔴	NO❗
└	swapExactETHForTokens	External ❗	💷	NO❗
└	swapTokensForExactETH	External ❗	🔴	NO❗
└	swapExactTokensForETH	External ❗	🔴	NO❗
└	swapETHForExactTokens	External ❗	💷	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗

IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🔘	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🔘	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🔘	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💷	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🔘	NO❗
IUniswapV2Factory	Interface				
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🔘	NO❗
└	setFeeTo	External ❗	🔘	NO❗
└	setFeeToSetter	External ❗	🔘	NO❗
IterableMapping	Library				
└	get	Internal 🔒		
└	getIndexOfKey	Internal 🔒		
└	getKeyAtIndex	Internal 🔒		
└	size	Internal 🔒		
└	set	Internal 🔒	🔘	
└	remove	Internal 🔒	🔘	
DividendPayingTokenOptionalInterface	Interface				
└	withdrawableDividendOf	External ❗		NO❗
└	withdrawnDividendOf	External ❗		NO❗
└	accumulativeDividendOf	External ❗		NO❗
DividendPayingTokenInterface	Interface				
└	dividendOf	External ❗		NO❗
└	distributeDividends	External ❗	💷	NO❗
└	withdrawDividend	External ❗	🔘	NO❗
SafeMathInt	Library				
└	mul	Internal 🔒		
└	div	Internal 🔒		

└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
SafeMathUint	Library				
└	toInt256Safe	Internal 🔒		
SafeMath	Library				
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
Context	Implementation				
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context			
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑		
IERC20	Interface				
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗

└	allowance	External ❗		NO❗
└	approve	External ❗	🔘	NO❗
└	transferFrom	External ❗	🔘	NO❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🔘	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🔘	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🔘	NO❗
└	transferFrom	Public ❗	🔘	NO❗
└	increaseAllowance	Public ❗	🔘	NO❗
└	decreaseAllowance	Public ❗	🔘	NO❗
└	_transfer	Internal 🔒	🔘		
└	_mint	Internal 🔒	🔘		
└	_burn	Internal 🔒	🔘		
└	_approve	Internal 🔒	🔘		
└	_beforeTokenTransfer	Internal 🔒	🔘		
└	_afterTokenTransfer	Internal 🔒	🔘		
DividendPayingToken	Implementation	ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
└	<Constructor>	Public ❗	🔘	ERC20
└	<Receive Ether>	External ❗	💸	NO❗
└	distributeDividends	Public ❗	💸	NO❗
└	withdrawDividend	Public ❗	🔘	NO❗
└	_withdrawDividendOfUser	Internal 🔒	🔘		
└	dividendOf	Public ❗		NO❗
└	withdrawableDividendOf	Public ❗		NO❗
└	withdrawnDividendOf	Public ❗		NO❗

└	accumulativeDividendOf	Public ❗			NO❗
└	_transfer	Internal 🔒	🛑		
└	_mint	Internal 🔒	🛑		
└	_burn	Internal 🔒	🛑		
└	_setBalance	Internal 🔒	🛑		
WyssCoin	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💸	NO❗
└	updateDividendTracker	Public ❗	🛑	onlyOwner
└	updateUniswapV2Router	Public ❗	🛑	onlyOwner
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	addToBlackList	External ❗	🛑	onlyOwner
└	claimStuckTokens	External ❗	🛑	onlyOwner
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setFees	External ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	setSwapTokens	External ❗	🛑	onlyOwner
└	setMaxBuy	External ❗	🛑	onlyOwner
└	enableTrading	External ❗	🛑	onlyOwner
└	setMaxSell	External ❗	🛑	onlyOwner
└	setMaxWallet	External ❗	🛑	onlyOwner
└	setBuyBackLimit	External ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑		
└	updateGasForProcessing	Public ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getClaimWait	External ❗			NO❗
└	getTotalDividendsDistributed	External ❗			NO❗
└	isExcludedFromFees	Public ❗			NO❗
└	withdrawableDividendOf	Public ❗			NO❗
└	dividendTokenBalanceOf	Public ❗			NO❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	getAccountDividendsInfo	External ❗			NO❗
└	getAccountDividendsInfoAtIndex	External ❗			NO❗
└	processDividendTracker	External ❗	🛑	NO❗
└	claim	External ❗	🛑	NO❗
└	getLastProcessedIndex	External ❗			NO❗

└	getNumberOfDividendTokenHolders	External ❗			NO❗
└	_transfer	Internal 🔒	🛑		
└	swapAndDistribute	Private 🔐	🛑		
└	swapAndSendToFee	Private 🔐	🛑		
└	swapAndLiquify	Private 🔐	🛑		
└	swapTokensForEth	Private 🔐	🛑		
└	swapETHForTokens	Private 🔐	🛑		
└	addLiquidity	Private 🔐	🛑		
└	swapAndSendDividends	Private 🔐	🛑		
TOKENDividendTracker	Implementation	Ownable, DividendPayingToken		
└	<Constructor>	Public ❗	🛑	DividendPayingToken
└	_transfer	Internal 🔒			
└	withdrawDividend	Public ❗			NO❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗			NO❗
└	getNumberOfTokenHolders	External ❗			NO❗
└	getAccount	Public ❗			NO❗
└	getAccountAtIndex	Public ❗			NO❗
└	canAutoClaim	Private 🔐			
└	setBalance	External ❗	🛑	onlyOwner
└	process	Public ❗	🛑		NO❗
└	processAccount	Public ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Summary of the Audit

According to automatically test, the customer`s solidity smart contract is **Secured**.

The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

The test found 0 critical, 0 high, 0 medium, 3 low issues, and 2 notes.