Smart Contract Security Audit V1

Optimum Token

13/3/2022

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**: Binance Smart Chain
- **Contract Address**: 0xdFA13D1b2850f623ADD9921337e9193Ad3559f80
- **Code Source:**

https://bscscan.com/address/0xdFA13D1b2850f623ADD9921337e9193Ad3559f80#code

Token Information

- Name: Opx
- Total Supply: 100,000,000
- Holders:
- Total transactions:

Contracts address deployed to test net (BSC)

Optimum smart contract on testnet.bsc by the auditor to test every function (BSC Test Net)

https://testnet.bscscan.com/address/0x93cab231bac64debe6e8bbb4bec1edde859afb04

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**.

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 0 critical, 0 high, 0 medium, 3 low, 0 very low-level issues and 2 notes in all solidity files of the contract

The files:

Optimum.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
optimum.sol	**28f6f124630859f1db88e9b35ed7af9b157daf3266175c0a903ef81910ba346c**	0xdFA13D1b2850f623ADD9921337e9193Ad3559f80

- Contract: optimum
- Inherit: ERC20, Ownable
- 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**
Owner	✓	Read / public	**Passed**
_isBlacklisted	✓	Read / private	**Passed**
_marketingWalletAddress	✓	Read / private	**Passed**
automatedMarketMakerPairs	✓	Read / public	**Passed**

deadWallet	✓	Read / public	**Passed**
dividendTokenBalanceOf	✓	Read / public	**Passed**
dividendTracker	✓	Read / public	**Passed**
gasForProcessing	✓	Read / public	**Passed**
getAccountDividendsInfo	✓	Read / public	**Passed**
getAccountDividendsInfoAtIndex	✓	Read / public	**Passed**
getClaimWait	✓	Read / public	**Passed**
getLastProcessedIndex	✓	Read / public	**Passed**
getNumberOfDividendTokenHolders	✓	Read / public	**Passed**
getTotalDividendsDistributed	✓	Read / public	**Passed**
isExcludedFromFees	✓	Read / public	**Passed**
LiquidityFeeBuy	✓	Read / public	**Passed**
MarketingFeeBuy	✓	Read / public	**Passed**
swapTokensAtAmount	✓	Read / public	**Passed**
BUSDRewardsFeeSell	✓	Read / public	**Passed**
uniswapV2Pair	✓	Read / public	**Passed**
uniswapV2Router	✓	Read / public	**Passed**
BUSDRewardsFeeBuy	✓	Read / public	**Passed**
BUSD	✓	Read / public	**Passed**
withdrawableDividendOf	✓	Read / public	**Passed**
sellLockTime	✓	Read / public	**Passed**
sellLockDisabled	✓	Read / public	**Passed**
ROUTER_ADDRESS	✓	Read / public	**Passed**
ownersFeeSell	✓	Read / public	**Passed**
ownersFeeBuy	✓	Read / public	**Passed**
owner	✓	Read / public	**Passed**
MarketingFeeSell	✓	Read / public	**Passed**
LiquidityFeeSell	✓	Read / public	**Passed**

approve	✓	Write / public	**Passed**
TransferFrom	✓	Write / public	**Passed**
blacklistAddress	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
excludeFromDividends	✓	Write / public	**Passed**
claim	✓	Write / public	**Passed**
updateDividendTracker	✓	Write / public	**Passed**
excludeFromFees	✓	Write / public	**Passed**
excludeMultipleAccounts FromFees	✓	Write / public	**Passed**
DisableSellLock	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setAutomatedMarketMak erPair	✓	Write / public	**Passed**
setSellLockTime	✓	Write / public	**Passed**
setLiquiditFee	✓	Write / public	**Passed**
setMarketingFee	✓	Write / public	**Passed**
setMarketingWallet	✓	Write / public	**Passed**
setSwapTokensAtAmount	✓	Write / public	**Passed**
setBUSDRewardsFee	✓	Write / public	**Passed**
processDividendTracker	✓	Write / public	**Passed**
updateClaimWait	✓	Write / public	**Passed**
updateDividendTracker	✓	Write / public	**Passed**
updateGasForProcessing	✓	Write / public	**Passed**
updateUniswapV2Router	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Design Logic.	**Passed**
6	Timestamp dependence.	**Passed**
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**

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:

No critical severity vulnerabilities were found.

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

The owner can change Buy and Sell Fees (Liquidity, BUSD Reward, and Marketing) or make it = zero.
The owner can add any address to Blacklist.
The owner can include / exclude any address from Fees or Reward.
The owner can set Sell Lock Time or exclude any address from it or disable it for all.

```
function DisableSellLock(bool disabled) public onlyOwner{
        sellLockDisabled=disabled;}
     function setSellLockTime(uint256 sellLockSeconds)public onlyOwner{
        require(sellLockSeconds<= MaxsellLockTime,"exceeds MaxSellLockTime");
        sellLockTime=sellLockSeconds;}
function excludeFromsellLock( address Address) public onlyOwner{
        _excludedFromSellLock[Address] = true ; }
```

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
        require(_isExcludedFromFees[account] != excluded, "Optimum: Account is
already the value of 'excluded'");
        _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 setMarketingWallet(address payable wallet) external onlyOwner{
        _marketingWalletAddress = wallet;}
    function setBUSDRewardsFee(uint256 value) external onlyOwner{
        BUSDRewardsFee = value;
        totalFees =
BUSDRewardsFee.add(liquidityFee).add(PromotionFee).add(ownersFee);}
    function setLiquiditFee(uint256 value) external onlyOwner{
        liquidityFee = value;
        totalFees =
BUSDRewardsFee.add(liquidityFee).add(PromotionFee).add(ownersFee);}
    function setMarketingFee(uint256 value) external onlyOwner{
        PromotionFee = value;
        totalFees =
BUSDRewardsFee.add(liquidityFee).add(PromotionFee).add(ownersFee);}
```

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 0.6.12 instead of ^0.6.12). 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: Acknowledged.

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 “_marketingWalletAddress” 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 = 2000000 * (10**18);

    mapping(address => bool) public _isBlacklisted;
    uint256 _tTotal = 1000000000000 * (10**18) ;
```

Recommendation

Replace the initialization as

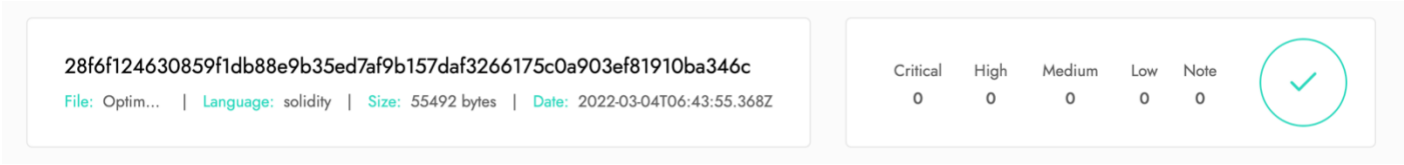
```
uint256 public swapTokensAtAmount = 2000000000000000000000000;

    mapping(address => bool) public _isBlacklisted;
    uint256 _tTotal = 1000000000000000000000000000000;
```

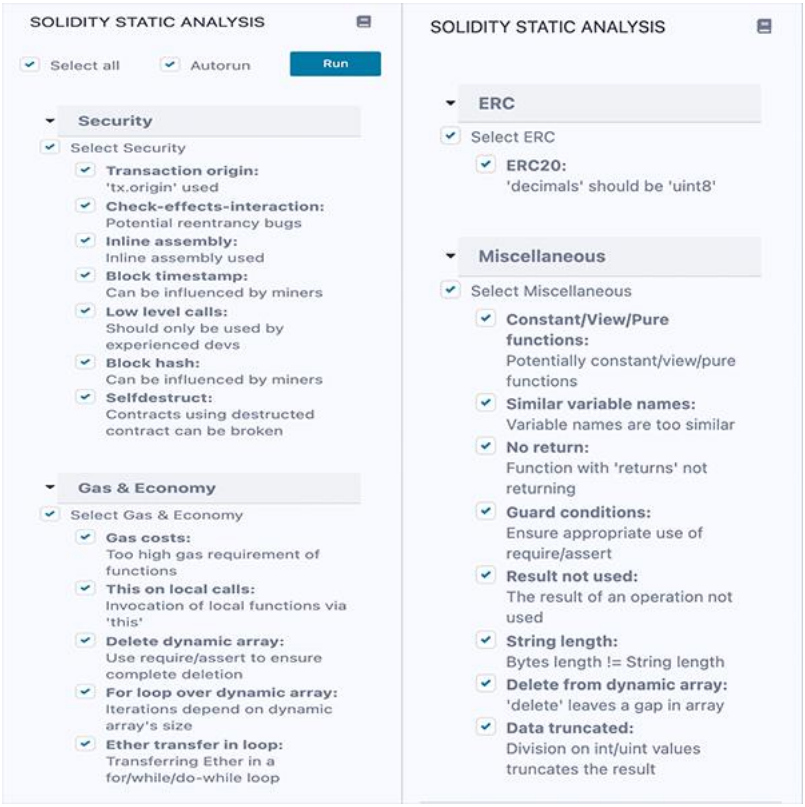
Status: Acknowledged

Automatic Testing

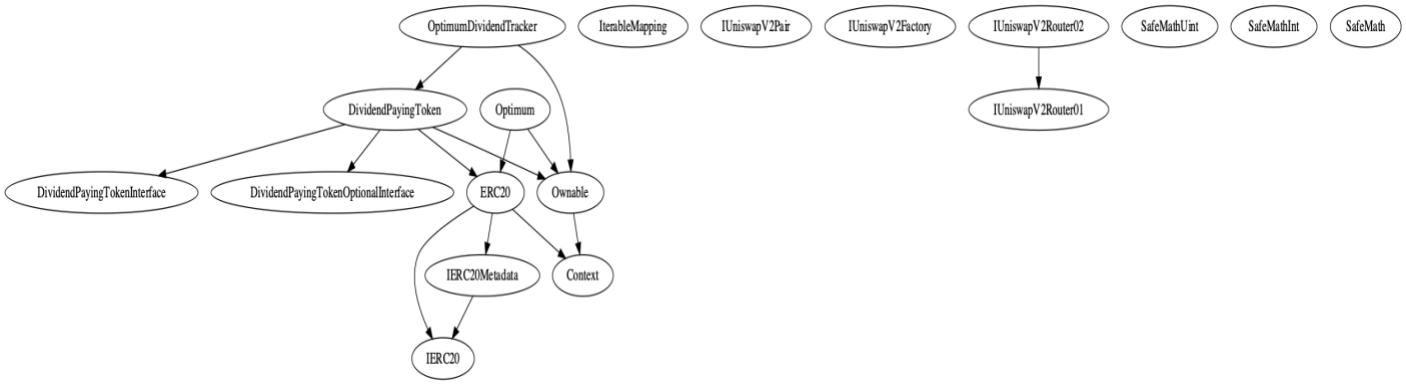
1- Check for security



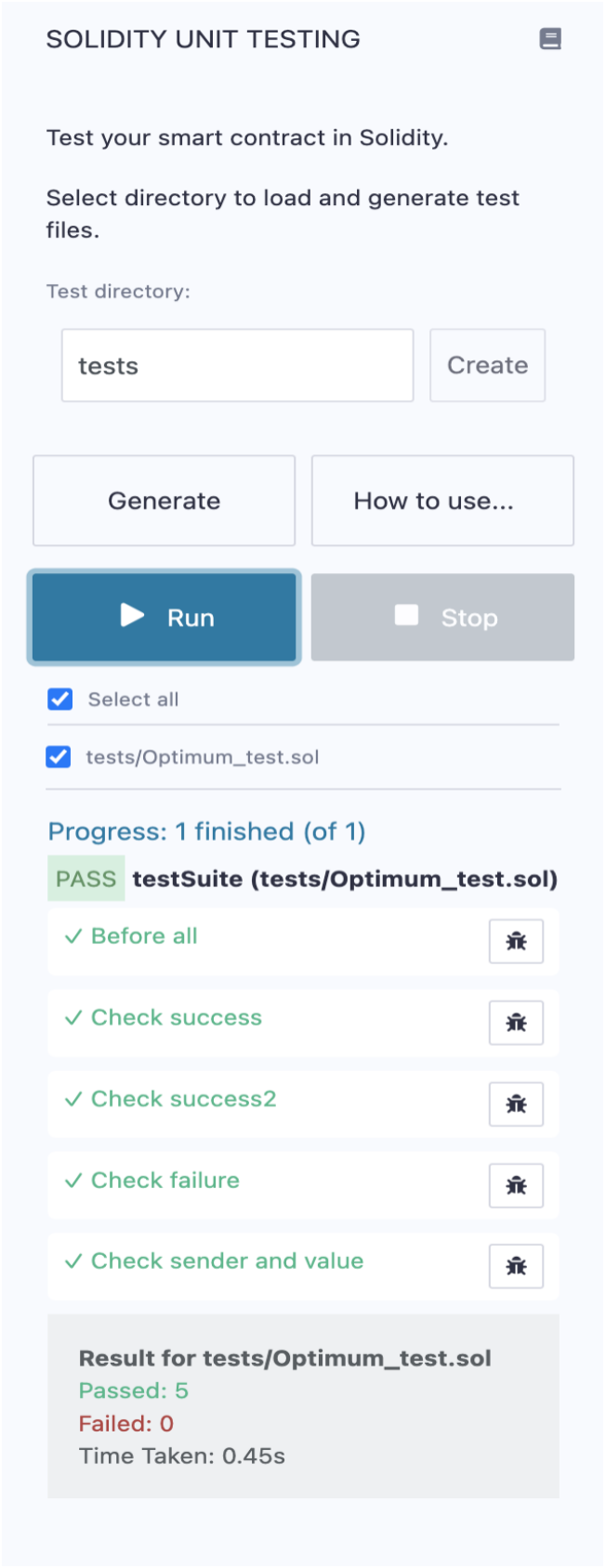
2- SOLIDITY STATIC ANALYSIS



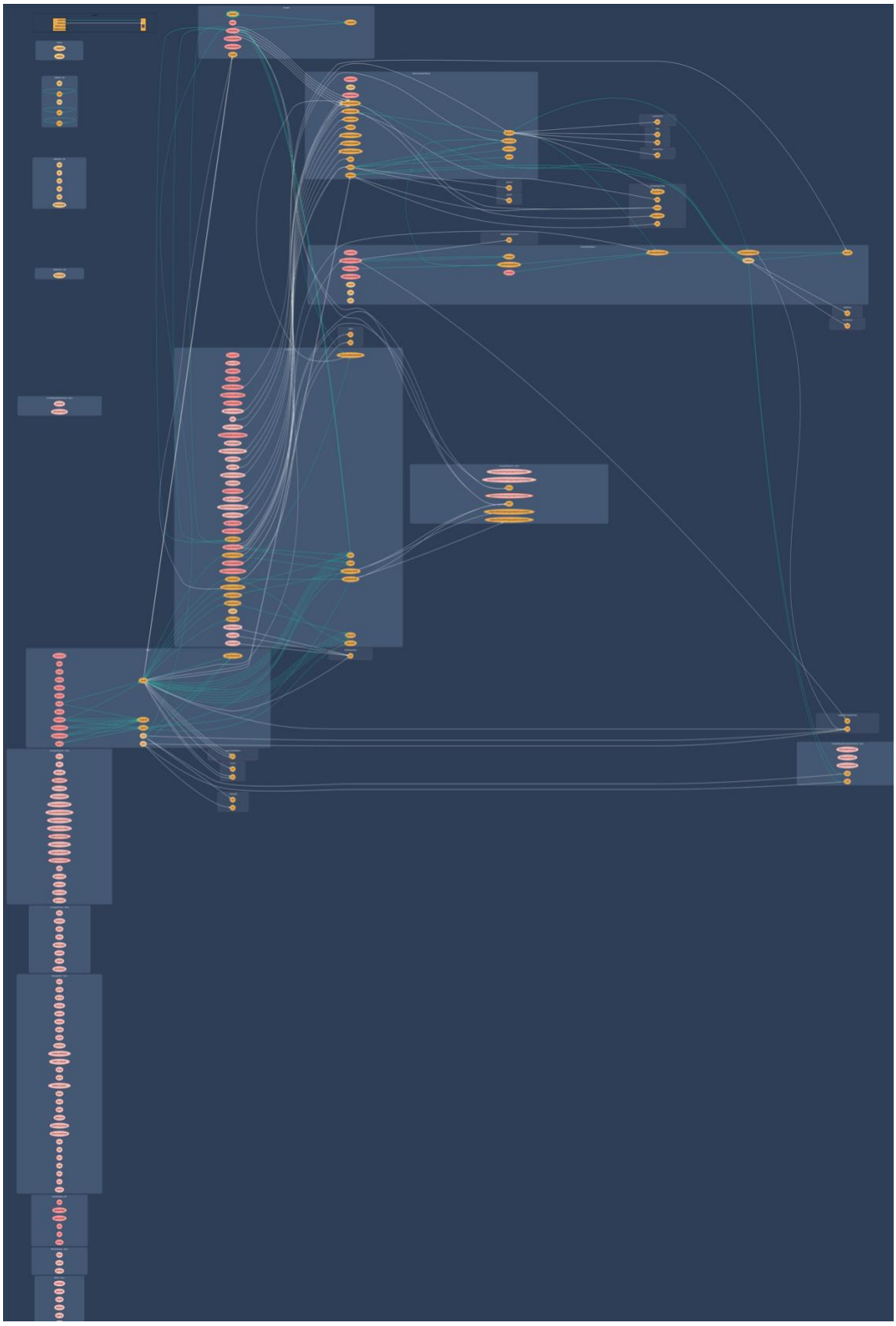
3- Inheritance graph



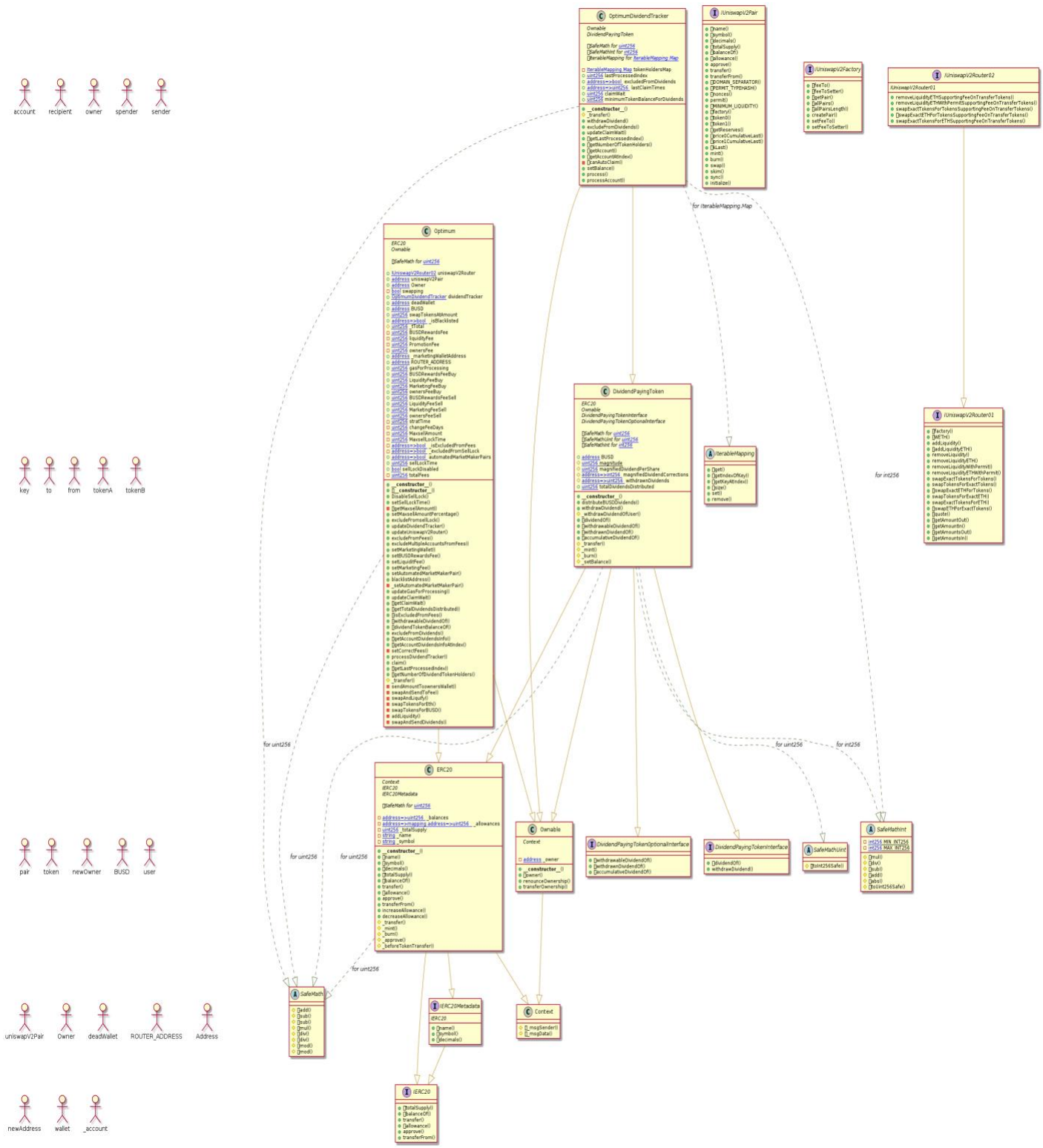
4- SOLIDITY UNIT TESTING



5- Call graph



Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
43509138  =>  div(int256,int256)
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()
268d8e2e  =>  get(Map,address)
b45dad3d  =>  getIndexOfKey(Map,address)
7596720f  =>  getKeyAtIndex(Map,uint256)
b1b533f3  =>  size(Map)
6b06f325  =>  set(Map,address,uint256)
0eac8729  =>  remove(Map,address)
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
c45a0155  =>  factory()
0dfe1681  =>  token0()
d21220a7  =>  token1()
0902f1ac  =>  getReserves()
5909c0d5  =>  price0CumulativeLast()
5a3d5493  =>  price1CumulativeLast()
7464fc3d  =>  kLast()
6a627842  =>  mint(address)
89afcb44  =>  burn(address)
022c0d9f  =>  swap(uint256,uint256,address,bytes)
bc25cf77  =>  skim(address)
fff6cae9  =>  sync()
485cc955  =>  initialize(address,address)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
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,b
ool,uint8,bytes32,bytes32)
ded9382a  =>
```

```
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,u
int8,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,add
ress,uint256)
5b0d5984  =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,u
int256,address,uint256,bool,uint8,bytes32,bytes32)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],add
ress,uint256)
b6f9de95  =>
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint25
6)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],addres
s,uint256)
91b89fba  =>  dividendOf(address)
6a474002  =>  withdrawDividend()
e823b9bf  =>  toInt256Safe(uint256)
bbe93d91  =>  mul(int256,int256)
adefc37b  =>  sub(int256,int256)
a5f3c23b  =>  add(int256,int256)
1b5ac4b5  =>  abs(int256)
744f7c7d  =>  toUint256Safe(int256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
f43f523a  =>  mod(uint256,uint256)
71af23e8  =>  mod(uint256,uint256,string)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
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)
a8b9d240  =>  withdrawableDividendOf(address)
aafd847a  =>  withdrawnDividendOf(address)
27ce0147  =>  accumulativeDividendOf(address)
40c2c69d  =>  distributeBUSDDividends(uint256)
```

```
373de4aa  =>  _withdrawDividendOfUser(address)
ab86e0a6  =>  _setBalance(address,uint256)
a9146d18  =>  DisableSellLock(bool)
4956569e  =>  setSellLockTime(uint256)
04a5b008  =>  getMaxsellAmount(address)
cee9a4c2  =>  setMaxsellAmountPercentage(uint256)
f24920e5  =>  excludeFromsellLock(address)
88bdd9be  =>  updateDividendTracker(address)
65b8dbc0  =>  updateUniswapV2Router(address)
c0246668  =>  excludeFromFees(address,bool)
c492f046  =>  excludeMultipleAccountsFromFees(address[],bool)
5d098b38  =>  setMarketingWallet(address)
c9d37d97  =>  setBUSDRewardsFee(uint256)
adefd90c  =>  setLiquiditFee(uint256)
625e764c  =>  setMarketingFee(uint256)
9a7a23d6  =>  setAutomatedMarketMakerPair(address,bool)
455a4396  =>  blacklistAddress(address,bool)
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)
81bea110  =>  setCorrectFees(bool)
700bb191  =>  processDividendTracker(uint256)
4e71d92d  =>  claim()
e7841ec0  =>  getLastProcessedIndex()
64b0f653  =>  getNumberOfDividendTokenHolders()
13a11329  =>  sendAmountToownersWallet(address,uint256)
a210621e  =>  swapAndSendToFee(uint256)
173865ad  =>  swapAndLiquify(uint256)
b28805f4  =>  swapTokensForEth(uint256)
6a5f7f7c  =>  swapTokensForBUSD(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/Optimum.sol	3831596376f56fd1f444bda3c6d1f0b196831325

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
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❗
IterableMapping	Library			
└	get	Public ❗		NO❗
└	getIndexOfKey	Public ❗		NO❗
└	getKeyAtIndex	Public ❗		NO❗
└	size	Public ❗		NO❗
└	set	Public ❗	🛑	NO❗
└	remove	Public ❗	🛑	NO❗
IUniswapV2Pair	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🛑	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗


└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	mint	External ❗	🔴	NO❗
└	burn	External ❗	🔴	NO❗
└	swap	External ❗	🔴	NO❗
└	skim	External ❗	🔴	NO❗
└	sync	External ❗	🔴	NO❗
└	initialize	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❗
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❗
DividendPayingTokenInterface	Interface			
└	dividendOf	External ❗		NO❗
└	withdrawDividend	External ❗	🔴	NO❗
SafeMathUint	Library			
└	toInt256Safe	Internal 🔒		

SafeMathInt	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Context	Implementation		
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
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 🔒	🛑	
DividendPayingTokenOptionalInterface	Interface			
└	withdrawableDividendOf	External ❗		NO❗
└	withdrawnDividendOf	External ❗		NO❗
└	accumulativeDividendOf	External ❗		NO❗
DividendPayingToken	Implementation	ERC20, Ownable, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
└	<Constructor>	Public ❗	🛑	ERC20
└	distributeBUSDDividends	Public ❗	🛑	onlyOwner
└	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 🔒	🛑	
Optimum	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💵	NO❗
└	DisableSellLock	Public ❗	🛑	onlyOwner
└	setSellLockTime	Public ❗	🛑	onlyOwner
└	getMaxsellAmount	Private 🔐		
└	setMaxsellAmountPercentage	Public ❗	🛑	onlyOwner
└	excludeFromsellLock	Public ❗	🛑	onlyOwner
└	updateDividendTracker	Public ❗	🛑	onlyOwner
└	updateUniswapV2Router	Public ❗	🛑	onlyOwner
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setBUSDRewardsFee	External ❗	🛑	onlyOwner
└	setLiquiditFee	External ❗	🛑	onlyOwner
└	setMarketingFee	External ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	blacklistAddress	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❗
└	setCorrectFees	Private 🔐	🛑	
└	processDividendTracker	External ❗	🛑	NO❗
└	claim	External ❗	🛑	NO❗
└	getLastProcessedIndex	External ❗		NO❗
└	getNumberOfDividendTokenHolders	External ❗		NO❗
└	_transfer	Internal 🔒	🛑	
└	sendAmountToownersWallet	Private 🔐	🛑	
└	swapAndSendToFee	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	swapTokensForBUSD	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	swapAndSendDividends	Private 🔐	🛑	
OptimumDividendTracker	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

Conclusion

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

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.