SMART CONTRACT AUDIT REPORT

For

KingSanta (KingSanta)

Prepared By: SFI Team

Prepared for: KingSanta team

Prepared on: 2/12/2021

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.

- # Overview of the audit

The project has 1 file. It contains approx 741 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.

- # Attacks made to the contract

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.

- # Good things in smart contract

Compiler version is static: -
=> In this file, you have put “pragma solidity 0.7.4;” which is a good way to define the compiler version.

```
pragma solidity 0.7.4;
```

- **SafeMath library: -**

KingSanta is using SafeMath library it is a good thing. It protects the contract from overflow and underflow.

```
library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }
    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure
returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }
    function div(uint256 a, uint256 b, string memory errorMessage) internal pure
returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't
hold

        return c;
    }
}
```

- **BEP20 standard interface library : -**
  - Here you KingSanta token using BEP20 standard interface

```
                            interface IBEP20 {
    function totalSupply() external view returns
(uint256);

    function decimals() external view returns (uint8);

function symbol() external view returns (string memory);
    function name() external view returns (string
memory);

function getOwner() external view returns (address);

function balanceOf(address account) external view returns
(uint256);

function transfer(address recipient, uint256 amount)
external returns (bool);

function allowance(address _owner, address spender)
external view returns (uint256);

function approve(address spender, uint256 amount)
external returns (bool);

function transferFrom(address sender, address recipient,
uint256 amount) external returns (bool);
    event Transfer(address indexed from, address indexed
to, uint256 value);
    event Approval(address indexed owner, address indexed
spender, uint256 value);
}
```

  - Here you KingSanta token using contract Auth Allows for contract ownership along with multi-address authorization

```
abstract contract Auth {
 address internal owner;
 mapping (address => bool) internal authorizations;

 constructor(address _owner) {
     owner = _owner;
     authorizations[_owner] = true;
 }

 modifier onlyOwner() {
     require(isOwner(msg.sender), "!OWNER"); _;
 }
```

```
    modifier authorized() {
        require(isAuthorized(msg.sender), "!AUTHORIZED"); _;
    }

    function authorize(address adr) public onlyOwner {
        authorizations[adr] = true;
    }
    function unauthorize(address adr) public onlyOwner {
        authorizations[adr] = false;
    }
```

- Here you KingSanta token using IDEX Interfaces libraries (IDEXFactory, IDEXRouter, and IDividendDistributor)

```
interface IDEXFactory {
    function createPair(address tokenA, address tokenB) external returns (address
pair);
}

interface IDEXRouter {
    function factory() external pure returns (address);
    function WETH() external pure returns (address);
interface IDividendDistributor {
    function setDistributionCriteria(uint256 _minPeriod, uint256 _minDistribution)
external;
    function setShare(address shareholder, uint256 amount) external;
    function deposit() external payable;
    function process(uint256 gas) external;
}
```

- ## Critical vulnerabilities found in the contract

  There not Critical severity vulnerabilities found

- ## High vulnerabilities found in the contract

  There not High severity vulnerabilities found

- ## Medium vulnerabilities found in the contract

  There not Medium severity vulnerabilities found

- ## Low vulnerabilities found in the contract

  There not Low severity vulnerabilities found

- ## V. Low vulnerabilities found in the contract

Block timestamp:

```
block.timestamp
```

In detail
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.

o Notes

#ERC20:

```
function decimals() external view returns (uint8);
function decimals() external pure override returns (uint8) { return _decimals; }
```

In detail
ERC20 contract's "decimals" function should have "uint8" as return type

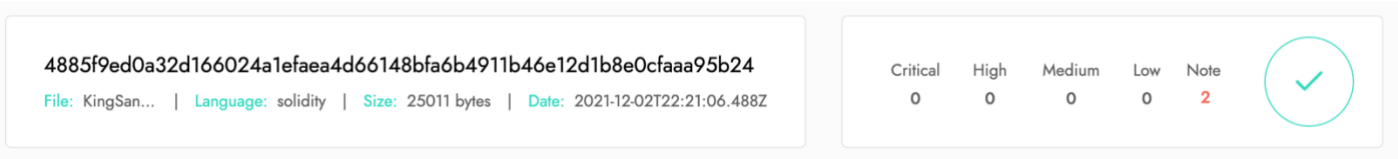
#Gas Costs:

```
function setDistributionCriteria(uint256 _minPeriod, uint256 _minDistribution)
external override onlyToken {
        minPeriod = _minPeriod;
        minDistribution = _minDistribution;
    }
```

In detail
Gas requirement of function KingSanta.setDistributionCriteria is infinite:If the gas requirement of a function is higher than the block gas limit, it cannot be executed Please avoid loops in your functions or actions that modify large areas of storage
(This includes clearing or copying arrays in storage)

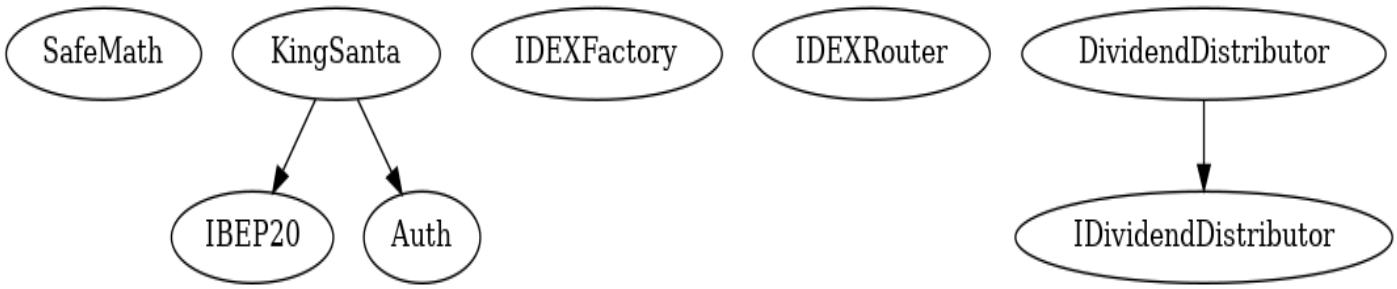
Testing proves:

1- Check for security

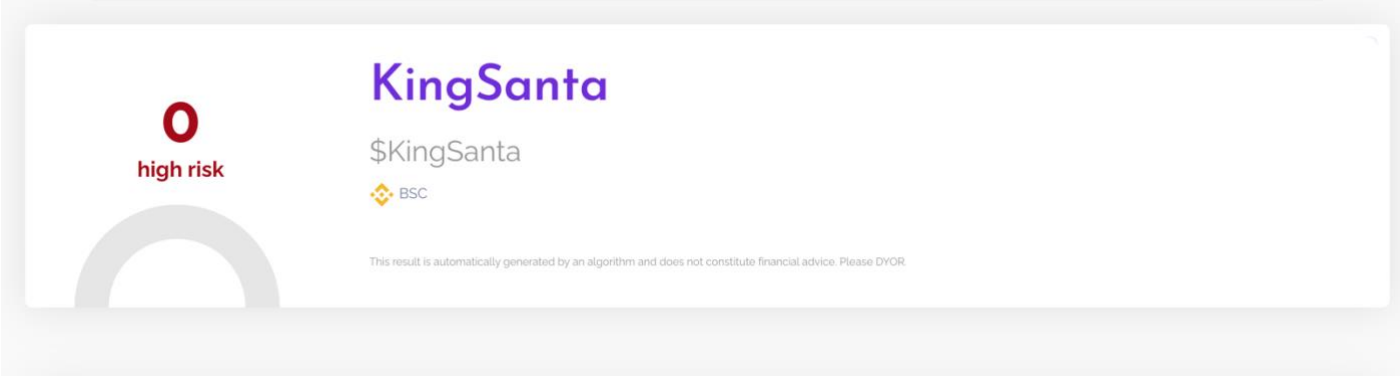


2- SOLIDITY STATIC ANALYSIS

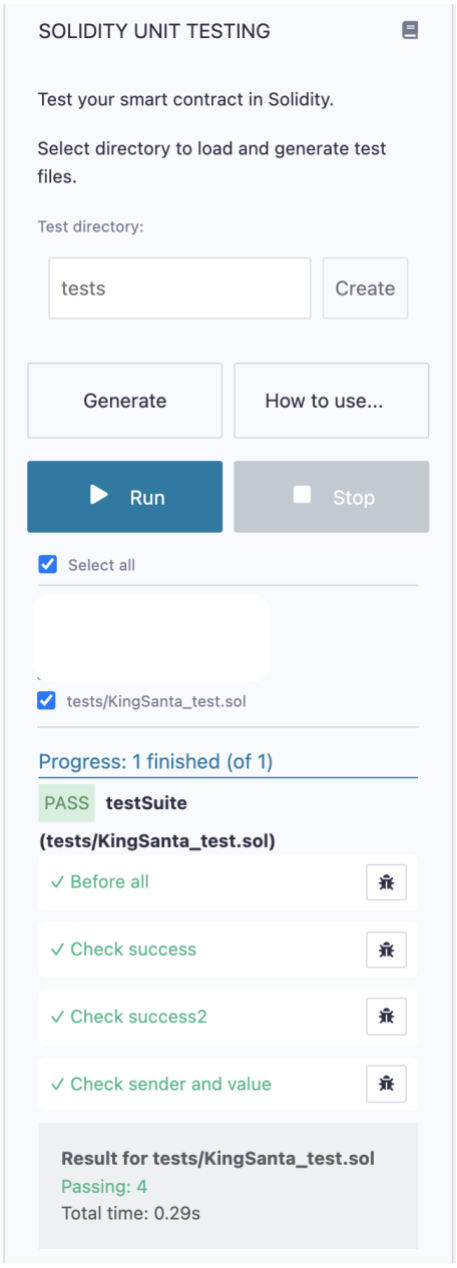
3- Inheritance graph



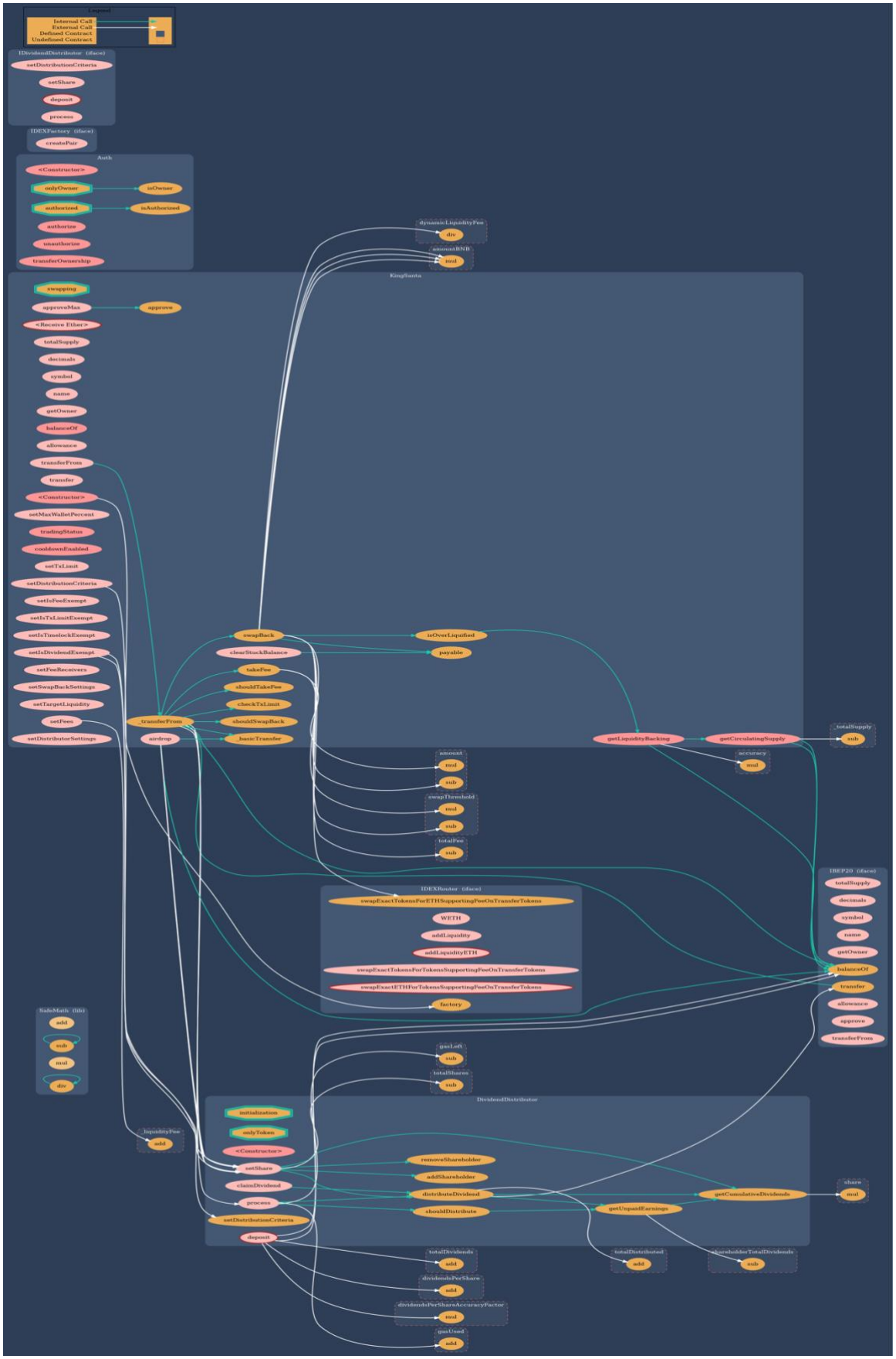
4- Solidity security scanner



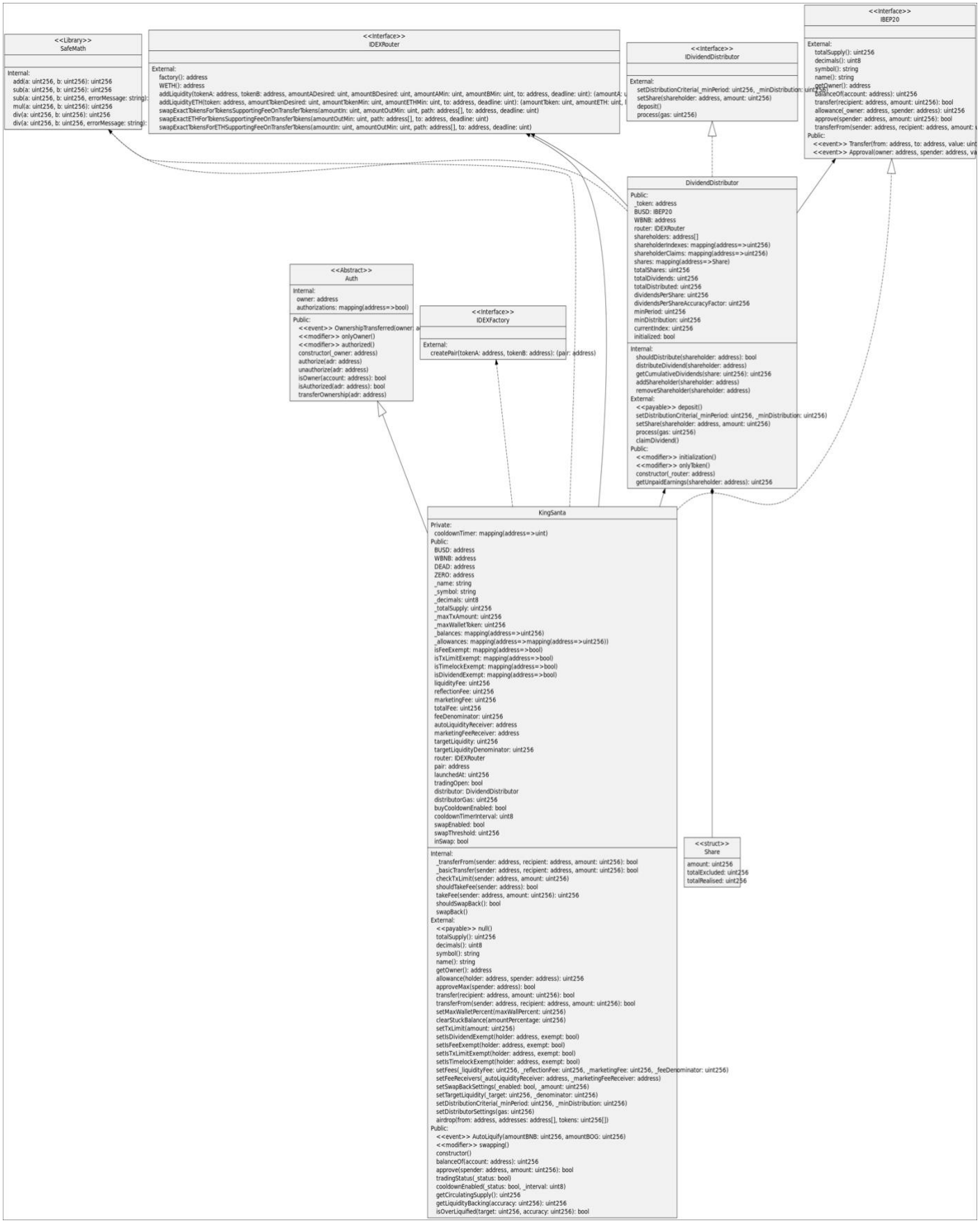
5- SOLIDITY UNIT TESTING



6- Call graph


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
dynamicLiquidityFee
div
amountBNB
mul
KingSanta
swapping
approveMax
approve
<Receive Ether>
totalSupply
decimals
symbol
name
getOwner
balanceOf
allowance
transferFrom
transfer
<Constructor>
setMaxWalletPercent
tradingStatus
cooldownEnabled
setTxLimit
setDistributionCriteria
setIsFeeExempt
setIsTxLimitExempt
setIsTimelockExempt
setIsDividendExempt
setFeeReceivers
setSwapBackSettings
setTargetLiquidity
setFees
setDistributorSettings
swapBack
clearStuckBalance
isOverLiquified
payable
takeFee
shouldTakeFee
checkTxLimit
transferFrom
shouldSwapBack
airdrop
basicTransfer
getLiquidityBacking
getCirculatingSupply
totalSupply
sub
accuracy
mul
amount
mul
sub
swapThreshold
mul
sub
totalFee
sub
IDEXRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
addLiquidity
addLiquidityETH
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
factory
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
add
sub
mul
div
gasLeft
sub
totalShares
sub
DividendDistributor
initialization
onlyToken
<Constructor>
setShare
claimDividend
process
setDistributionCriteria
deposit
removeShareholder
addShareholder
distributeDividend
shouldDistribute
getUnpaidEarnings
getCumulativeDividends
share
mul
totalDistributed
add
totalDividends
add
dividendsPerShare
add
dividendsPerShareAccuracyFactor
mul
gasUsed
add
shareholderTotalDividends
sub
liquidityFee
add

Unified Modeling Language (UML)



Function Signature

```
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)
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()
ad5c4648  =>  WETH()
e8e33700  =>
addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],add
ress,uint256)
b6f9de95  =>
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint25
6)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],addres
s,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)
82bf293c  =>  setMaxWalletPercent(uint256)
cb712535  =>  _transferFrom(address,address,uint256)
f0774e71  =>  _basicTransfer(address,address,uint256)
4afa518a  =>  checkTxLimit(address,uint256)
e7c44c69  =>  shouldTakeFee(address)
1d759aae  =>  takeFee(address,uint256)
0d5c6cea  =>  shouldSwapBack()
1da1db5e  =>  clearStuckBalance(uint256)
```

```
0d295980  =>  tradingStatus(bool)
2d594567  =>  cooldownEnabled(bool,uint8)
6ac5eeee  =>  swapBack()
5c85974f  =>  setTxLimit(uint256)
f708a64f  =>  setIsDividendExempt(address,bool)
658d4b7f  =>  setIsFeeExempt(address,bool)
f84ba65d  =>  setIsTxLimitExempt(address,bool)
50db71fb  =>  setIsTimelockExempt(address,bool)
6fcba377  =>  setFees(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)
025ff12f  =>  airdrop(address,address[],uint256[])
```

- # Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/KingSanta.sol	1e6d96f3d2c9c6f5f729ce219c39c98ee10721b4

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	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❗	
└	WETH	External ❗	NO❗	
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💸	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💸	NO❗

└	swapExactTokensForETHSupportingFeeOnTransferTokens	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 🔒	🛑	
KingSanta	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❗
└	setMaxWalletPercent	External ❗	🛑	onlyOwner
└	_transferFrom	Internal 🔒	🛑	
└	_basicTransfer	Internal 🔒	🛑	
└	checkTxLimit	Internal 🔒		
└	shouldTakeFee	Internal 🔒		
└	takeFee	Internal 🔒	🛑	
└	shouldSwapBack	Internal 🔒		
└	clearStuckBalance	External ❗	🛑	onlyOwner
└	tradingStatus	Public ❗	🛑	onlyOwner
└	cooldownEnabled	Public ❗	🛑	onlyOwner
└	swapBack	Internal 🔒	🛑	swapping
└	setTxLimit	External ❗	🛑	authorized
└	setIsDividendExempt	External ❗	🛑	authorized
└	setIsFeeExempt	External ❗	🛑	authorized
└	setIsTxLimitExempt	External ❗	🛑	authorized
└	setIsTimelockExempt	External ❗	🛑	authorized
└	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❗
└	airdrop	External ❗	🛑	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, 0 low, 1 Very low issues, and 2 notes.