SMART CONTRACT AUDIT REPORT

For

Vaccine Coin (VAX)

https://www.vaccinecoin.de/

Prepared By: SFI Team

Prepared for: VAX team

Prepared on: 8/12/2021

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 539 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 0.6.8;” which is a good way to define the compiler version.

```
pragma solidity 0.6.8;
```

- **SafeMath library: -**

VAX 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");
  }
```

- **Ownable library : -**
  - Here you VAX token using ownable library, Initializes the contract setting the deployer as the initial owner

```
  contract Ownable is Context {
  address private _owner;

  event OwnershipTransferred(address indexed previousOwner, address indexed
newOwner);
  constructor () internal {
    address msgSender = _msgSender();
    _owner = msgSender;
    emit OwnershipTransferred(address(0), msgSender);
  }


  function owner() public view returns (address) {
    return _owner;
  }


  modifier onlyOwner() {
    require(_owner == _msgSender(), "Ownable: caller is not the owner");
    _;
  }
```

- Here you VAX token using interface IBEP20 which Returns the amount of tokens in existence, symbol, name , owner and etc. based on IBEP20 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);
}
```

- ## 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

There not V. Low severity vulnerabilities found

- ## Notes

#ERC20:

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

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

#Gas Costs:

```
function transferOwnership(address newOwner) public onlyOwner {
    _transferOwnership(newOwner);
  }
string public _symbol;
  string public _name;
function symbol() external view virtual override returns (string memory) {
    return _symbol;
  }

  function name() external view virtual override returns (string memory) {
    return _name;
  }
function transfer(address recipient, uint256 amount) external override returns
(bool) {
    _transfer(_msgSender(), recipient, amount);
    return true;
  }
function approve(address spender, uint256 amount) external override returns (bool)
{
    _approve(_msgSender(), spender, amount);
    return true;
  }


  function transferFrom(address sender, address recipient, uint256 amount) external
override returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount,
"BEP20: transfer amount exceeds allowance"));
    return true;
  }
```

In detail
Gas requirement of function Vaccine Coin 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

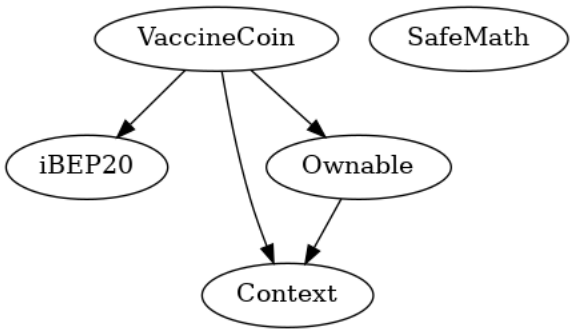
67e033c885c17bca09f40c1c64bdab0f490fafc521ba080911a275b13541c6b1

File: VaccineC... | Language: solidity | Size: 16928 bytes | Date: 2021-12-08T11:39:57.518Z

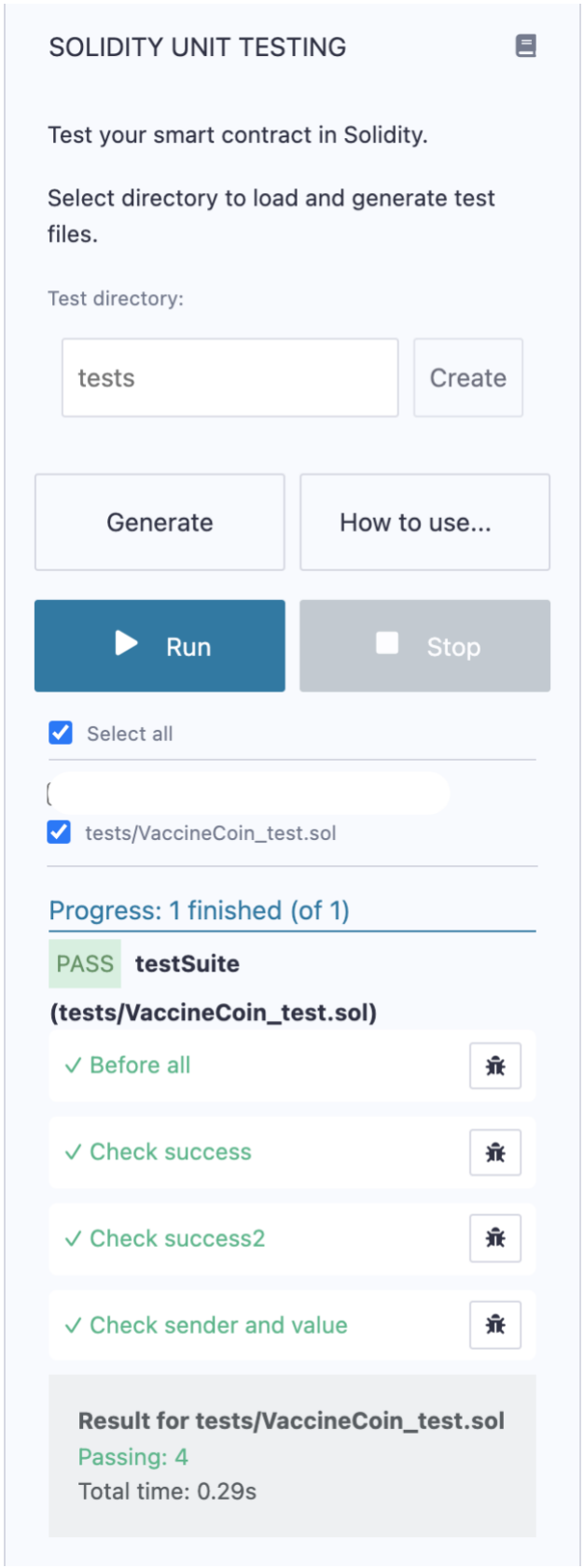
Critical	High	Medium	Low	Note
0	0	0	0	2

2- SOLIDITY STATIC ANALYSIS

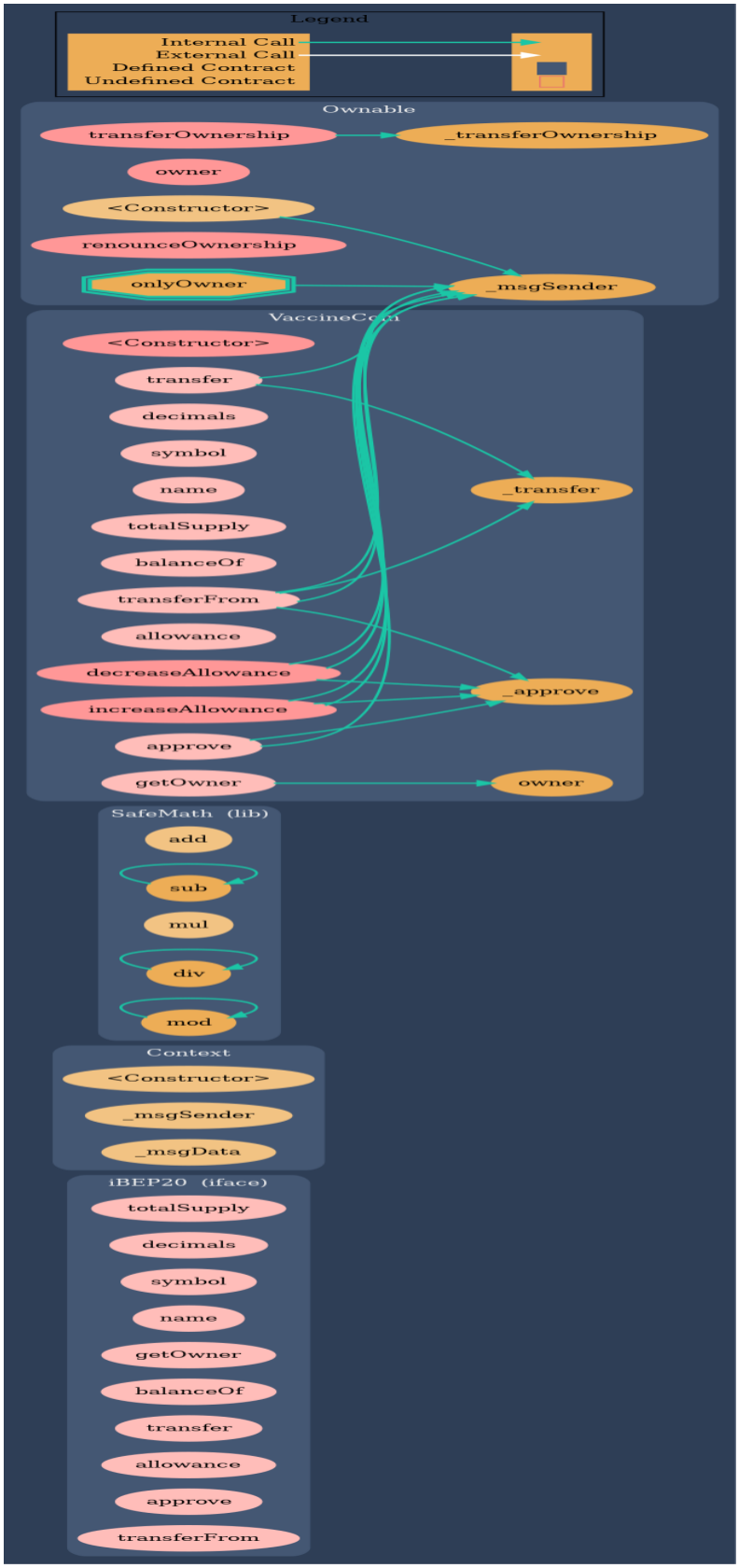
3- Inheritance graph



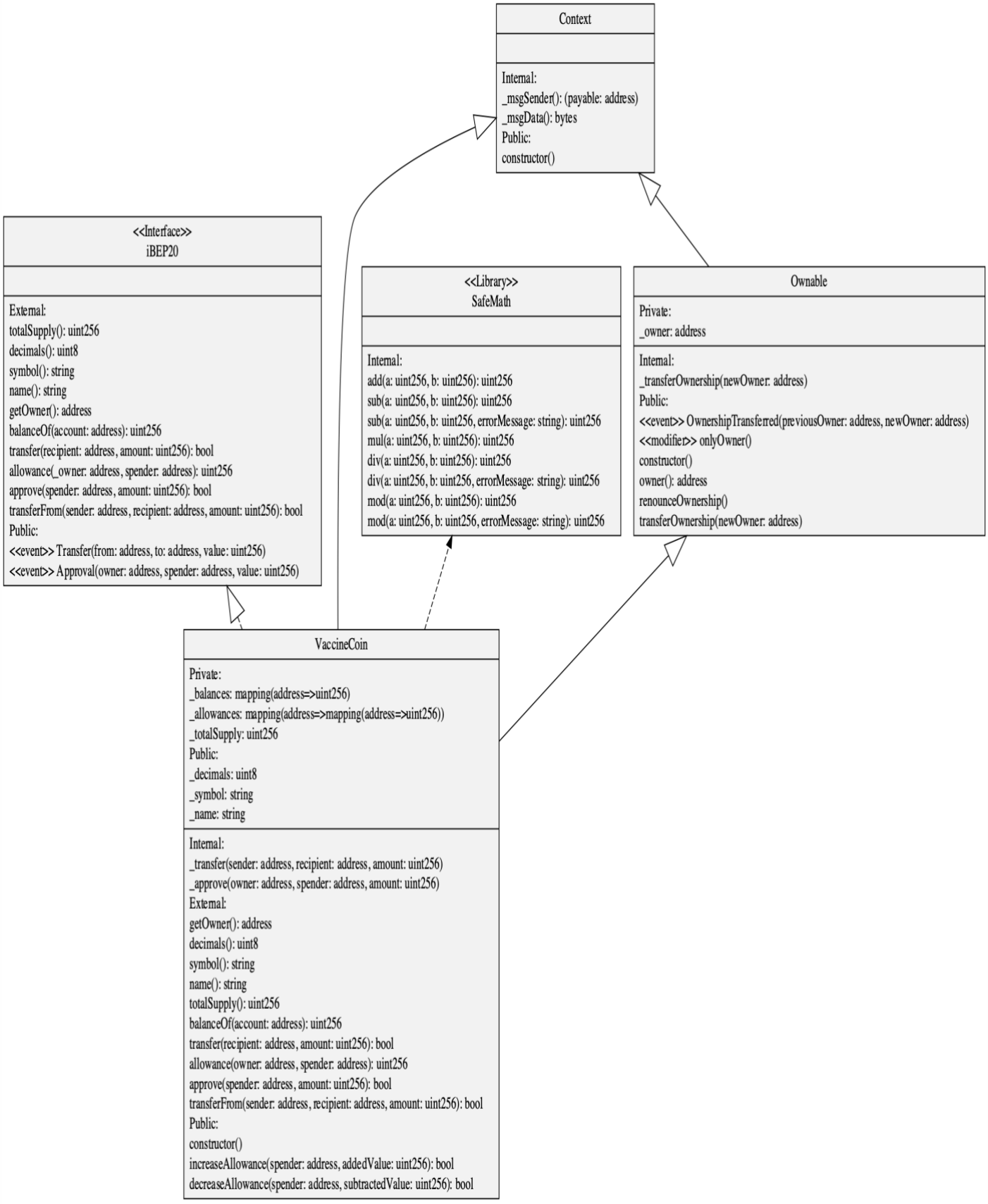
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
transferOwnership
_transferOwnership
owner
<Constructor>
renounceOwnership
onlyOwner
_msgSender
VaccineCoin
<Constructor>
transfer
decimals
symbol
name
totalSupply
balanceOf
transferFrom
allowance
decreaseAllowance
increaseAllowance
approve
getOwner
_transfer
_approve
owner
SafeMath (lib)
add
sub
mul
div
mod
Context
<Constructor>
_msgSender
_msgData
iBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom

Unified Modeling Language (UML)



Function Signature

```
39509351  =>  increaseAllowance(address,uint256)
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)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
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)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
104e81ff  =>  _approve(address,address,uint256)
```

- # Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/VaccineCoin.sol	e94ab6d0a00ce5d0a33a6c781207042752e2c462

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
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❗
Context	Implementation			
└	<Constructor>	Internal 🔒	🛑	
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
VaccineCoin	Implementation	Context, iBEP20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	getOwner	External ❗		NO❗
└	decimals	External ❗		NO❗

└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_transfer	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	

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, 0 Very low issues, and 2 notes.