SMART CONTRACT AUDIT REPORT

For

Food Bank (Food)

https://foodbankcrypto.org/

Prepared By: SFI Team

Prepared for: Food Bank team

Prepared on: 12/12/2021

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 861 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 0.6.12;” which is a good way to define the compiler version.

```
pragma solidity 0.6.12;
```

- **SafeMath library: -**

Food Bank 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) {
        require(b > 0, errorMessage);
        uint256 c = a / b;

        return c;
    }

    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        return mod(a, b, "SafeMath: modulo by zero");
    }

    function mod(uint256 a, uint256 b, string memory errorMessage) internal pure
returns (uint256) {
        require(b != 0, errorMessage);
        return a % b;
    }
}
```

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

```
                contract Ownable is Context {
    address private _owner;
    address private _previousOwner;
    uint256 private _lockTime;

    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");
        _;
    }

    function renounceOwnership() public virtual onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
```

```
emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }

    function geUnlockTime() public view returns (uint256) {
        return _lockTime;
    }

    function lock(uint256 time) public virtual onlyOwner {
        _previousOwner = _owner;
        _owner = address(0);
        _lockTime = now + time;
        emit OwnershipTransferred(_owner, address(0));
    }

    function unlock() public virtual {
        require(_previousOwner == msg.sender, "You don't have permission to
unlock");
        require(now > _lockTime , "Contract is locked until 7 days");
        emit OwnershipTransferred(_owner, _previousOwner);
        _owner = _previousOwner;
    }
}
```

- Here you Food Bank using IUNiswap Interfaces libraries (IUNiswap Factory, IUNiswap Router, and IUniswap Pair)

```
interface IUniswapV2Factory {
    event PairCreated(address indexed token0, address indexed token1, address pair,
unit);
interface IUniswapV2Pair {
    event Approval(address indexed owner, address indexed spender, uint value);
    event Transfer(address indexed from, address indexed to, uint value);

interface IUniswapV2Router01 {
    function factory() external pure returns (address);
    function WETH() external pure returns (address);
```

- **Critical vulnerabilities found in the contract**

  There are no Critical severity vulnerabilities found

- **High vulnerabilities found in the contract**

  There are no High severity vulnerabilities found

- **Medium vulnerabilities found in the contract**

  There are no Medium severity vulnerabilities found

- **Low vulnerabilities found in the contract**

  There are no Low severity vulnerabilities found

- **V. Low vulnerabilities found in the contract**

Block timestamp:

```
                function swapTokensForEth(uint256 tokenAmount) private {

        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();

        _approve(address(this), address(uniswapV2Router), tokenAmount);

        uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokenAmount,
            0,
            path,
            address(this),
            block.timestamp
        );
    }

    function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {

        _approve(address(this), address(uniswapV2Router), tokenAmount);
```

```
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0,
            0,
            owner(),
            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);
```

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

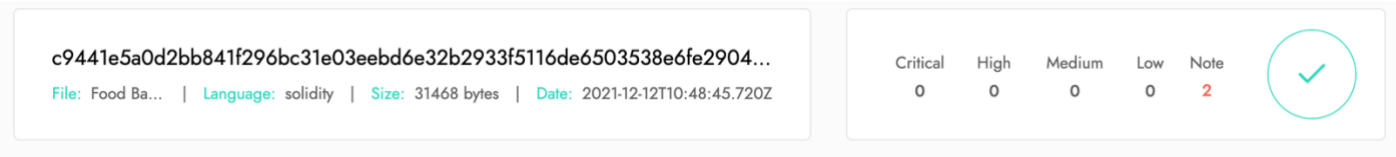
#Gas Costs:

```
function renounceOwnership() public virtual onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }
```

In detail
Gas requirement of function FoodBank.renounceOwnership 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)

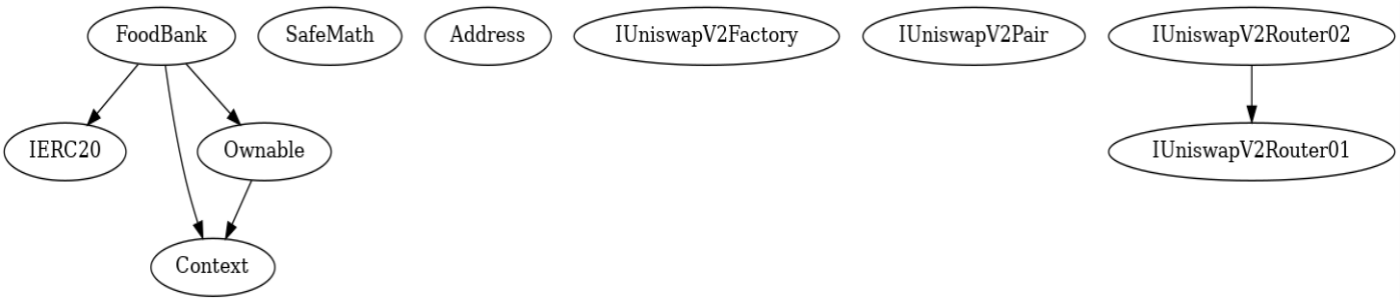
Testing proves:

1- Check for security



2- SOLIDITY STATIC ANALYSIS

3- Inheritance graph



4- SOLIDITY UNIT TESTING CODE & RESULTS

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

pragma solidity >=0.4.22 <0.9.0;

// This import is automatically injected by Truffle
import "Truffle _tests.sol";

// This import is required to use custom transaction context
// Although it may fail compilation in 'Solidity Compiler' plugin

import "Truffle_accounts.sol";
import "../Food Bank Token.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:
        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");
    }


    /// #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");
    }
}
```

SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

tests

Create

Generate

How to use...

Run

Stop

Select all

tests/Food Bank Token_test.sol

Progress: 1 finished (of 1)

testSuite (tests/Food Bank Token_test.sol)

✓ Before all

✓ Check success

✓ Check success2

✓ Check failure

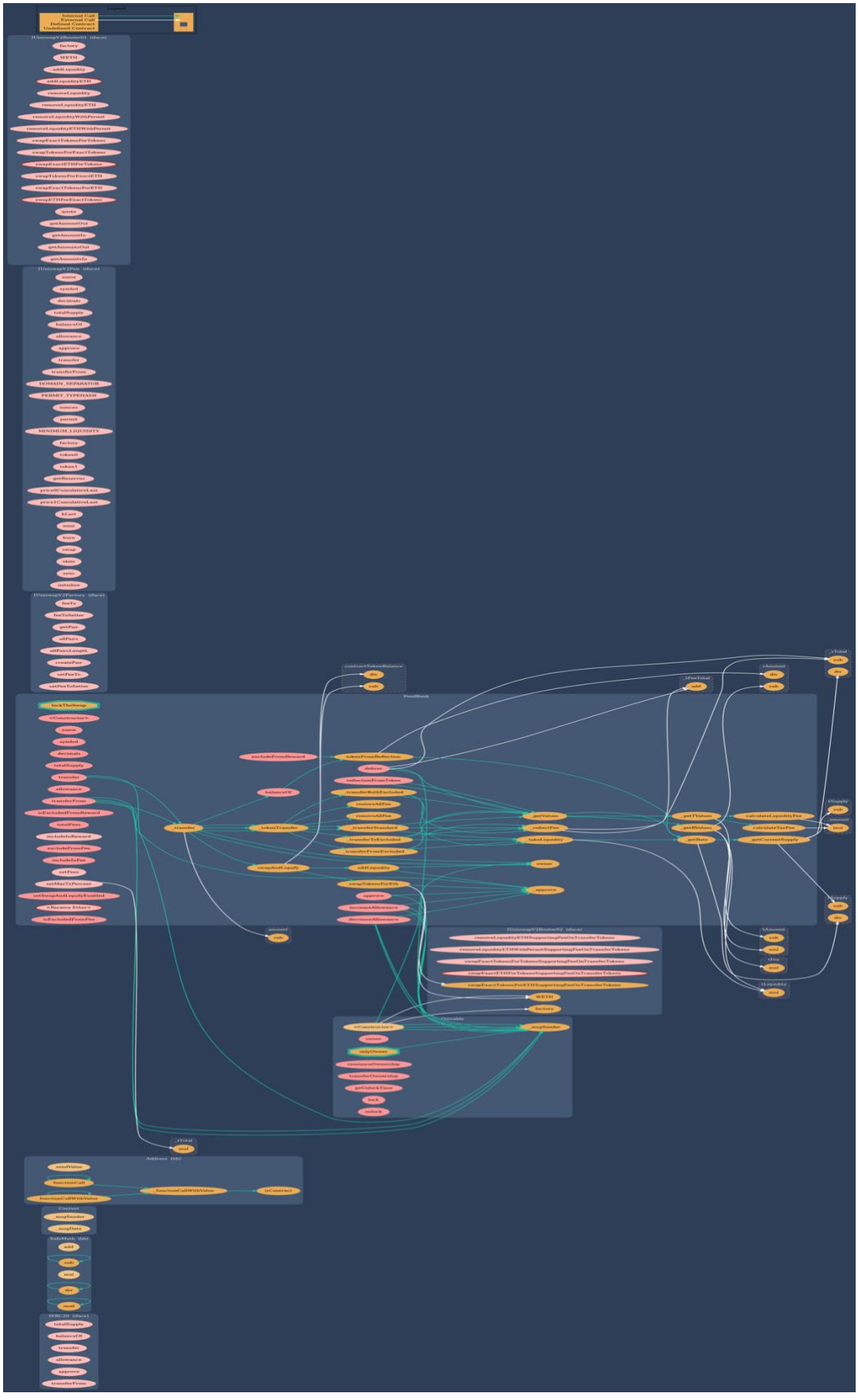
✓ Check sender and value

Result for tests/Food Bank Token_test.sol

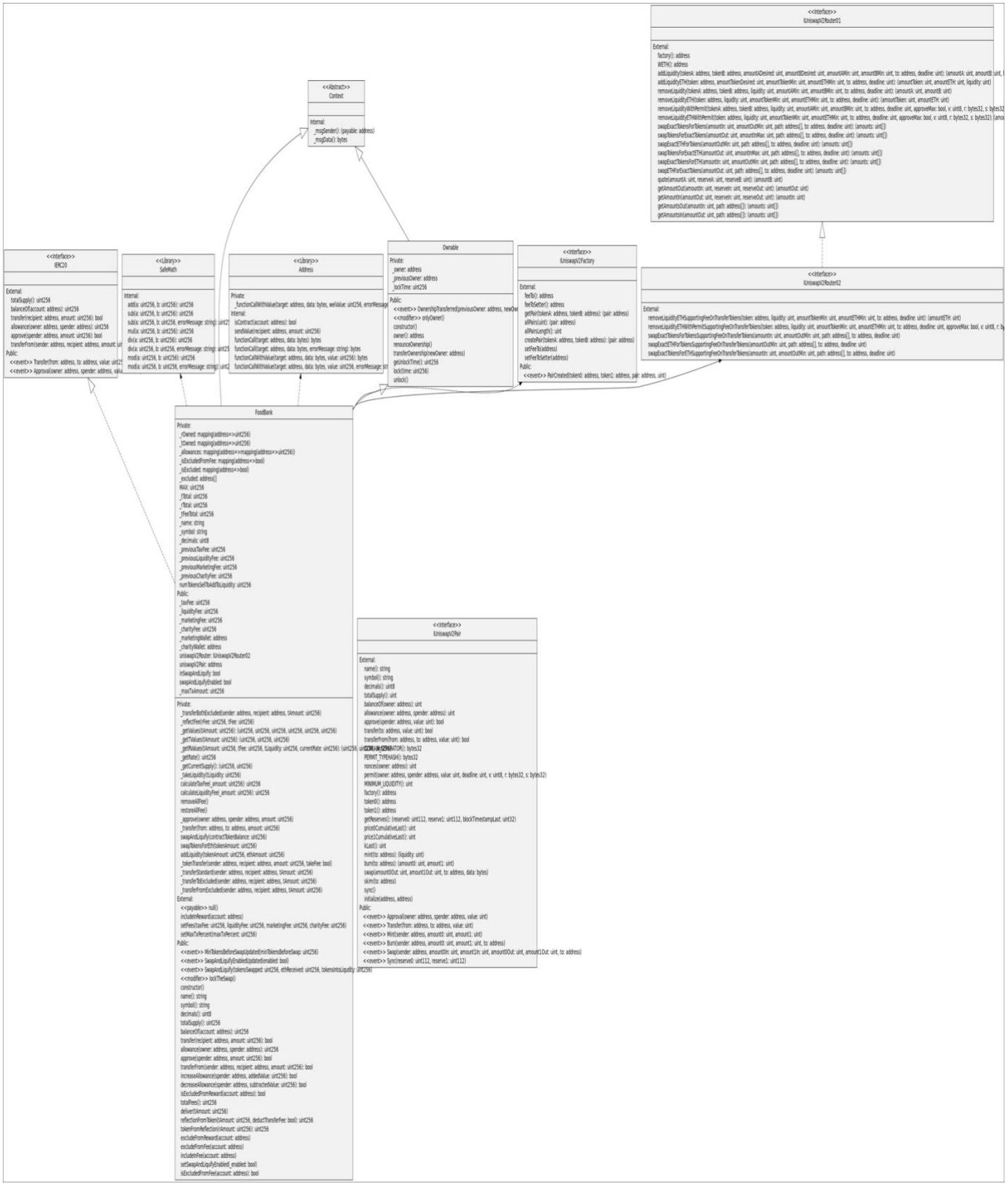
Passing: 5

Total time: 2.50s

5- Call graph



Unified Modeling Language (UML)



Function Signature

```
11902160  =>  _getTValues(uint256)
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
75128141  =>  calculateTaxFee(uint256)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
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()
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
36455e42  =>  _functionCallWithValue(address,bytes,uint256,string)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
b6c52324  =>  geUnlockTime()
dd467064  =>  lock(uint256)
a69df4b5  =>  unlock()
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
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)
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)
a457c2d7  =>  decreaseAllowance(address,uint256)
88f82020  =>  isExcludedFromReward(address)
13114a9d  =>  totalFees()
3bd5d173  =>  deliver(uint256)
4549b039  =>  reflectionFromToken(uint256,bool)
2d838119  =>  tokenFromReflection(uint256)
52390c02  =>  excludeFromReward(address)
3685d419  =>  includeInReward(address)
6ff6cdf4  =>  _transferBothExcluded(address,address,uint256)
437823ec  =>  excludeFromFee(address)
ea2f0b37  =>  includeInFee(address)
6fcba377  =>  setFees(uint256,uint256,uint256,uint256)
d543dbeb  =>  setMaxTxPercent(uint256)
c49b9a80  =>  setSwapAndLiquifyEnabled(bool)
184d894e  =>  _reflectFee(uint256,uint256)
```

```
d4780e36  =>  _getValues(uint256)
1d5671e4  =>  _getRValues(uint256,uint256,uint256,uint256)
94e10784  =>  _getRate()
97a9d560  =>  _getCurrentSupply()
c432df5e  =>  _takeLiquidity(uint256)
cc126a23  =>  calculateLiquidityFee(uint256)
301370af  =>  removeAllFee()
e7e3e3a7  =>  restoreAllFee()
5342acb4  =>  isExcludedFromFee(address)
104e81ff  =>  _approve(address,address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
173865ad  =>  swapAndLiquify(uint256)
b28805f4  =>  swapTokensForEth(uint256)
9cd441da  =>  addLiquidity(uint256,uint256)
b09bbc79  =>  _tokenTransfer(address,address,uint256,bool)
2852df65  =>  _transferStandard(address,address,uint256)
16f1cc83  =>  _transferToExcluded(address,address,uint256)
c7d9be66  =>  _transferFromExcluded(address,address,uint256)
```

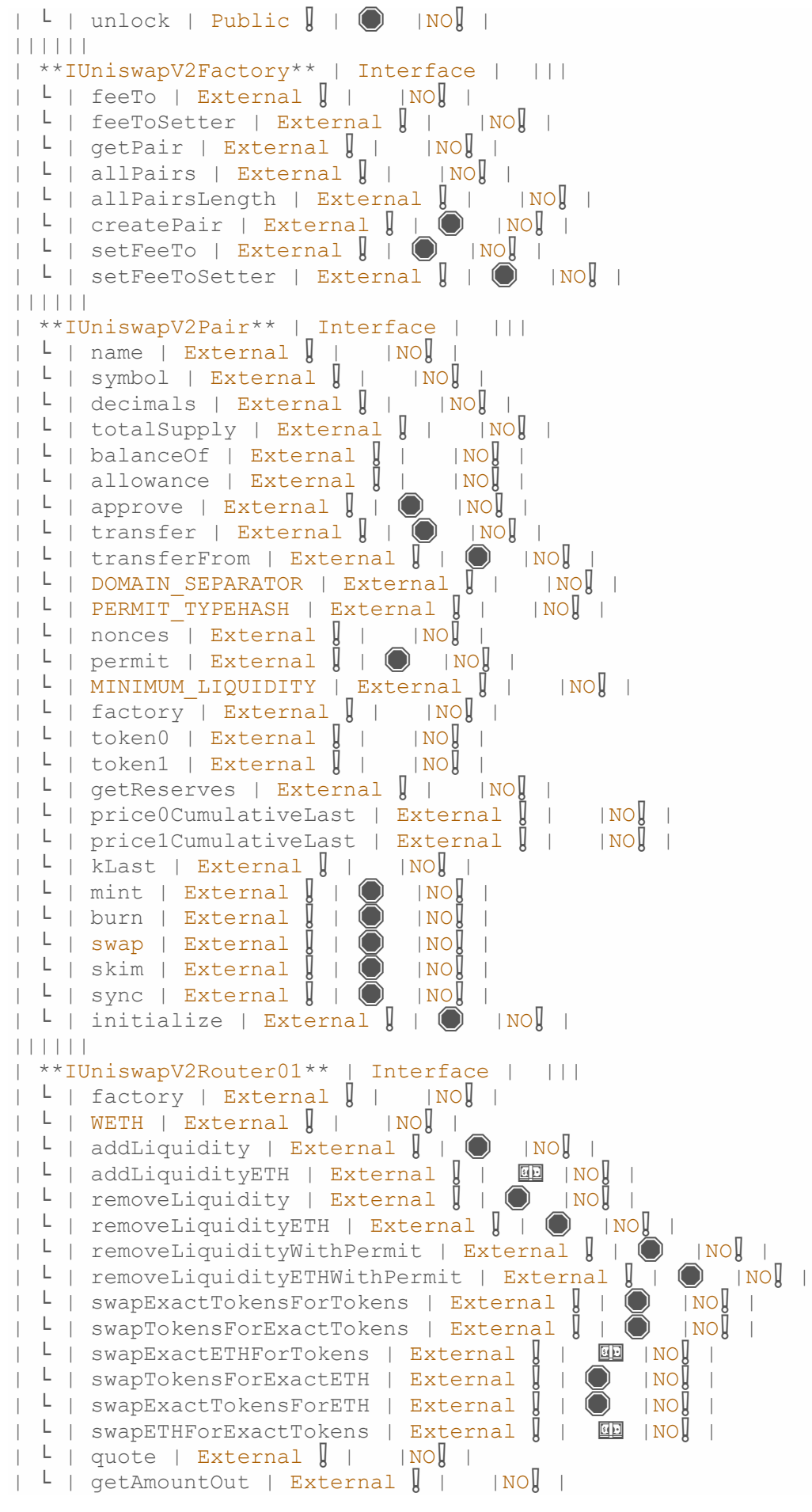
- # Automatic general report

files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Food Bank Token.sol	fe06aea14a1ba0a6bf28fd5a60d0f3a89cd6850c

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❗
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
Ownable	Implementation	Context		
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	geUnlockTime	Public ❗		NO❗
└	lock	Public ❗	🛑	onlyOwner


└	unlock	Public ❗	🔘	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❗
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❗
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❗
FoodBank	Implementation	Context, IERC20, Ownable		
└	<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❗
└	isExcludedFromReward	Public ❗		NO❗
└	totalFees	Public ❗		NO❗
└	deliver	Public ❗	🛑	NO❗
└	reflectionFromToken	Public ❗		NO❗
└	tokenFromReflection	Public ❗		NO❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	_transferBothExcluded	Private 🔐	🛑	
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	setFees	External ❗	🛑	onlyOwner
└	setMaxTxPercent	External ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💰	NO❗
└	_reflectFee	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_takeLiquidity	Private 🔐	🛑	
└	calculateTaxFee	Private 🔐		
└	calculateLiquidityFee	Private 🔐		
└	removeAllFee	Private 🔐	🛑	
└	restoreAllFee	Private 🔐	🛑	
└	isExcludedFromFee	Public ❗		NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	

└	_transferStandard	Private 🔐	🛑	
└	_transferToExcluded	Private 🔐	🛑	
└	_transferFromExcluded	Private 🔐	🛑	

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.