Smart Contract
Security Audit
V1

NANAX Token Smart Contract

27/8/2022

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**: 0x9A3C7F233a666026b5c90097309BdBB9c5561ad9
- **Code Source:**

https://bscscan.com/address/0x9A3C7F233a666026b5c90097309BdBB9c5561ad9#code

Token Information

- Name: **NANAX**
- Total Supply: 100,000,000
- Holders:
- Total transactions:

Contracts address deployed to test net (BSC)

NANAX Token smart contract on BSC test net by the auditor to test every function (BSC Test Net)

https://testnet.bscscan.com/address/0x55968143f37ef688ecc848fef5a666fc5edfd99c

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, 1 high, 0 medium, 3 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

NANAX.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
NANAX.sol	**5f334645ab34196e55b7ae24cb0c2ab6a2da311d890259f4c670d1f4123aef6b**	0x9A3C7F233a666026b5c90097309BdBB9c5561ad9

- Contract: NANAX
- 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**
isExcludedFromFees	✓	Read / public	**Passed**
marketingWallet	✓	Read / public	**Passed**
marketingFees	✓	Read / public	**Passed**
maxTransactionAmount	✓	Read / public	**Passed**
swapEnabled	✓	Read / public	**Passed**

swapTokensAtAmount	✓	Read / public	**Passed**
uniswapV2Router	✓	Read / public	**Passed**
uniswapV2Pair	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
TransferFrom	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**
setMarketingWallet	✓	Write / public	**Passed**
setFees	✓	Write / public	**Passed**
excludeFromFees	✓	Write / public	**Passed**
setMaxTransaction	✓	Write / public	**Passed**
setSwapEnabled	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setSwapTokensAtAmount	✓	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 with notes**
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:

# Logic errors

Description

The smart contract has 3% marketing fees should goes to marketing wallet but, after the auditor test this function found the fees goes to the token address not to the marketing wallet.

You can check this transaction:

https://testnet.bscscan.com/tx/0xa9ac9e0049fc3fb37377f04ce4f56889d5a17552a63d02091bb971000279704e

```
function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal override {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");

        if(amount == 0) {
            super._transfer(from, to, 0);
            return;
        }

        if(!_isExcludedFromFees[from] && !_isExcludedFromFees[to]) {
            require(amount <= maxTransactionAmount, "amount exceeds the 
maxTransactionAmount.");
        }

        uint256 contractTokenBalance = balanceOf(address(this));

        bool overMinTokenBalance = contractTokenBalance >= swapTokensAtAmount;

        if(
            overMinTokenBalance &&
            !inSwap &&
            to==uniswapV2Pair &&
            swapEnabled
        ) {
            contractTokenBalance = swapTokensAtAmount;
            swapTokensForBNB(contractTokenBalance, marketingWallet);
        }
         // if any account belongs to _isExcludedFromFee account then remove the 
fee
        if(!_isExcludedFromFees[from] && !_isExcludedFromFees[to] && 
(from==uniswapV2Pair || to==uniswapV2Pair)) {
            uint256 fees = amount.mul(marketingFee).div(100);
            if(fees > 0) {
                super._transfer(from, address(this), fees);
            }

            amount = amount.sub(fees);
        }

        super._transfer(from, to, amount);
    }
```

Remediation

Rewrite this function and make fees goes to marketing wallet and testing it.

Status: Closed. Fixed in version2.

Medium:

No Medium severity vulnerabilities were found.

Low:

#Missing zero address validation

Description

When the owner wants to change Marketing wallet, he has to check for the zero address to make, he didn't add the zero address. Otherwise, marketing fees will be burn fees.

```
function setMarketingWallet(address payable _newAddress) external onlyOwner() {
        marketingWallet = _newAddress;
    }
```

Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version2.

#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 enable or disable the trade.
The owner can include / exclude any address from Fees.
The owner can change the marketing fees.

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
        _isExcludedFromFees[account] = excluded;
        emit ExcludeFromFees(account, excluded);
    }
    function setSwapTokensAtAmount(uint256 newAmt) external onlyOwner() {
        swapTokensAtAmount = newAmt;
    }

    function setSwapEnabled(bool _enabled) public onlyOwner {
        swapEnabled = _enabled;
        emit SwapEnabledUpdated(_enabled);
    }
    function setFee(uint256 _newFee) public onlyOwner {
        require(_newFee <= 5, "tax tooo high");
        marketingFee = _newFee;
    }
```

Remediation

Make these functions internal in next version or the team should announce the investors before change anything and give them time if they want to change anything too.

P.S: This issue is common to the majority of rewards smart contracts.

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

# Constant calculations in the contract

Description

recalculated initialization will save 2847 units of gas in deployment

```
uint256 public maxTransactionAmount = 100000000 * (10**18);
    uint256 public swapTokensAtAmount = 20000 * (10**18);
_createTSupply(owner(), 100000000*10**18);
```

Recommendation

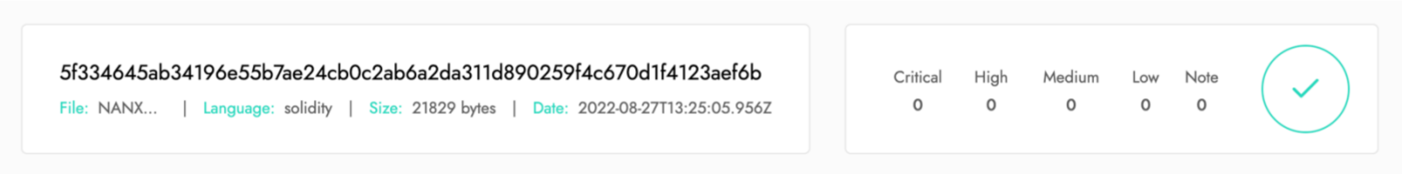
Replace the initialization as

```
uint256 public maxTransactionAmount = 100000000000000000000000000;
    uint256 public swapTokensAtAmount = 20000000000000000000000;
_createTSupply(owner(), 100000000000000000000000000);
```

Status Acknowledged.

Automatic Testing

1- Check for security

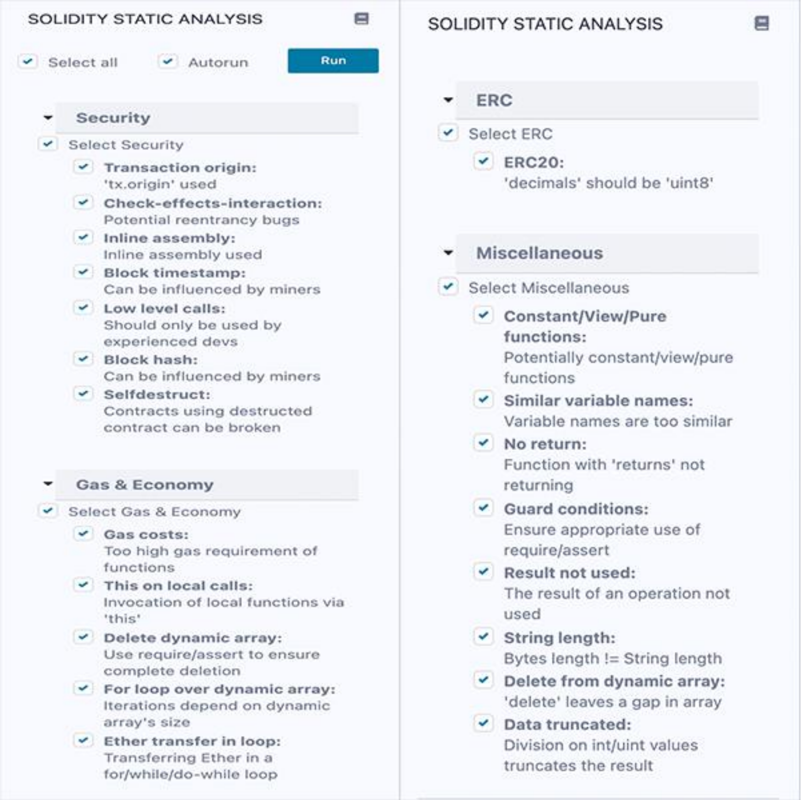


5f334645ab34196e55b7ae24cb0c2ab6a2da311d890259f4c670d1f4123aef6b

File: NANX... | Language: solidity | Size: 21829 bytes | Date: 2022-08-27T13:25:05.956Z

Critical	High	Medium	Low	Note
0	0	0	0	0

2- SOLIDITY STATIC ANALYSIS



SOLIDITY STATIC ANALYSIS

☑ Select all ☑ Autorun Run

Security

☑ Select Security

☑ **Transaction origin:** 'tx.origin' used

☑ **Check-effects-interaction:** Potential reentrancy bugs

☑ **Inline assembly:** Inline assembly used

☑ **Block timestamp:** Can be influenced by miners

☑ **Low level calls:** Should only be used by experienced devs

☑ **Block hash:** Can be influenced by miners

☑ **Selfdestruct:** Contracts using destructed contract can be broken

Gas & Economy

☑ Select Gas & Economy

☑ **Gas costs:** Too high gas requirement of functions

☑ **This on local calls:** Invocation of local functions via 'this'

☑ **Delete dynamic array:** Use require/assert to ensure complete deletion

☑ **For loop over dynamic array:** Iterations depend on dynamic array's size

☑ **Ether transfer in loop:** Transferring Ether in a for/while/do-while loop

SOLIDITY STATIC ANALYSIS

ERC

☑ Select ERC

☑ **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

☑ Select Miscellaneous

☑ **Constant/View/Pure functions:** Potentially constant/view/pure functions

☑ **Similar variable names:** Variable names are too similar

☑ **No return:** Function with 'returns' not returning

☑ **Guard conditions:** Ensure appropriate use of require/assert

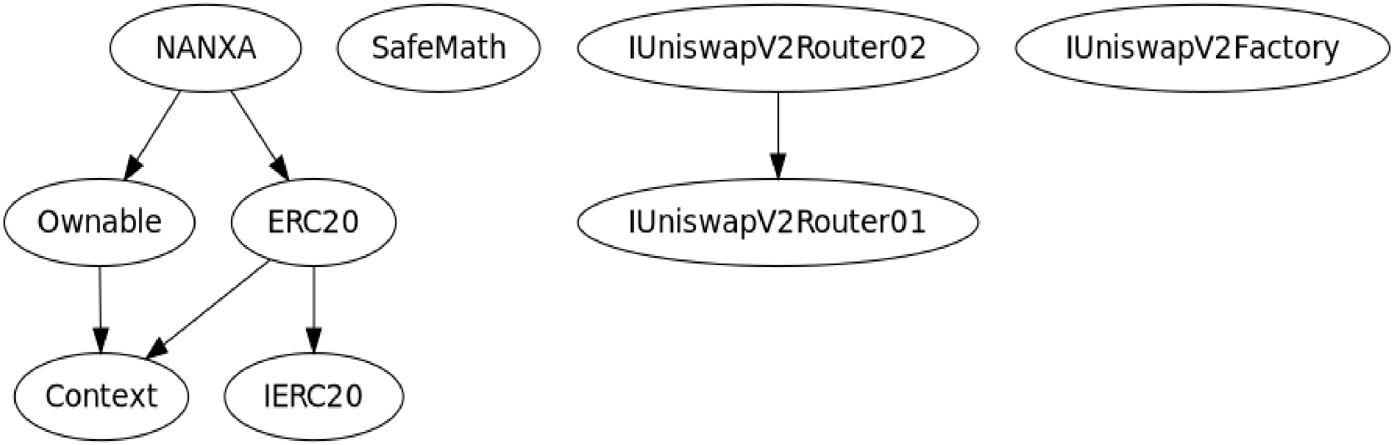
☑ **Result not used:** The result of an operation not used

☑ **String length:** Bytes length != String length

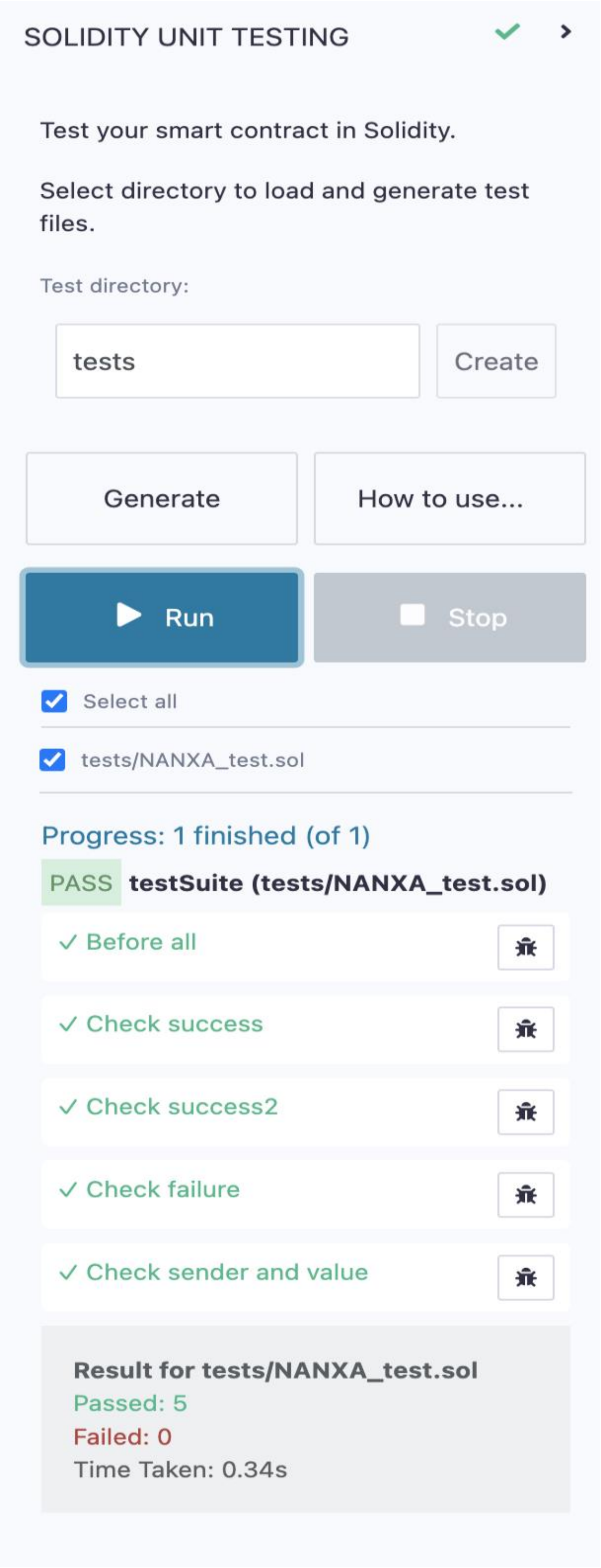
☑ **Delete from dynamic array:** 'delete' leaves a gap in array

☑ **Data truncated:** Division on int/uint values truncates the result

3- Inheritance graph



4- SOLIDITY UNIT TESTING



Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory: tests

☑ Select all

☑ tests/NANXA_test.sol

Progress: 1 finished (of 1)

PASS **testSuite (tests/NANXA_test.sol)**

✓ Before all

✓ Check success

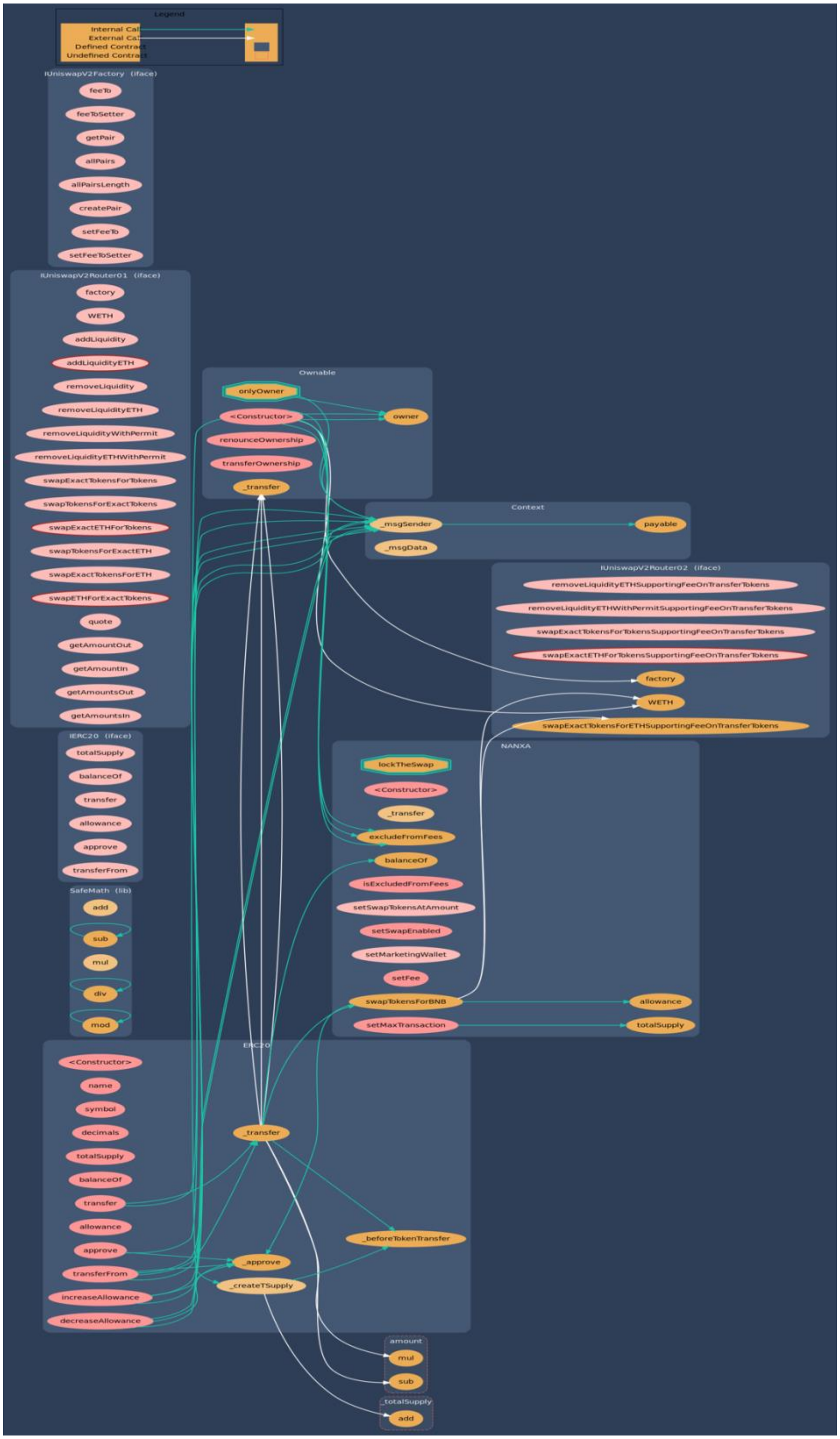
✓ Check success2

✓ Check failure

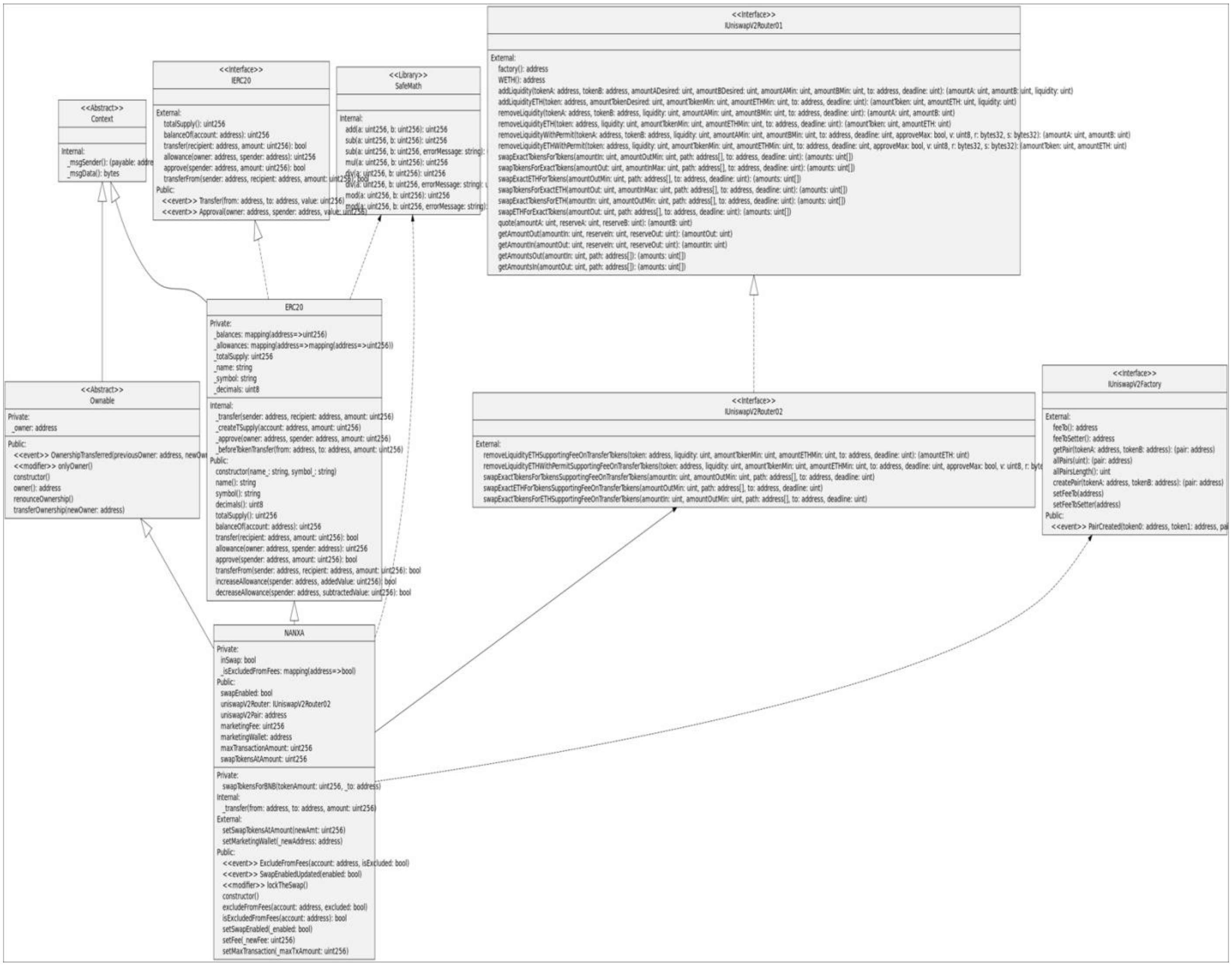
✓ Check sender and value

Result for tests/NANXA_test.sol
Passed: 5
Failed: 0
Time Taken: 0.34s

5- Call graph


Legend
Internal Cal
External Ca
Defined Contract
Undefined Contract
IUniswapV2Factory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
add
sub
mul
div
mod
Ownable
onlyOwner
<Constructor>
renounceOwnership
transferOwnership
_transfer
owner
Context
_msgSender
_msgData
payable
IUniswapV2Router02 (iface)
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
factory
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
NANXA
lockTheSwap
<Constructor>
_transfer
excludeFromFees
balanceOf
isExcludedFromFees
setSwapTokensAtAmount
setSwapEnabled
setMarketingWallet
setFee
swapTokensForBNB
allowance
setMaxTransaction
totalSupply
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
increaseAllowance
decreaseAllowance
_transfer
_beforeTokenTransfer
_approve
_createTSupply
amount
mul
sub
_totalSupply
add

Unified Modeling Language (UML)



```
<<Abstract>>
Context

Internal:
  _msgSender(): (payable: addre
  _msgData(): bytes
```

```
<<Interface>>
IERC20

External:
  totalSupply(): uint256
  balanceOf(account: address): uint256
  transfer(recipient: address, amount: uint256): bool
  allowance(owner: address, spender: address): uint256
  approve(spender: address, amount: uint256): bool
  transferFrom(sender: address, recipient: address, amount: uint256): bool
Public:
  <<event>> Transfer(from: address, to: address, value: uint256)
  <<event>> Approval(owner: address, spender: address, value: uint256)
```

```
<<Library>>
SafeMath

Internal:
  add(a: uint256, b: uint256): uint256
  sub(a: uint256, b: uint256): uint256
  sub(a: uint256, b: uint256, errorMessage: string):
  mul(a: uint256, b: uint256): uint256
  div(a: uint256, b: uint256): uint256
  div(a: uint256, b: uint256, errorMessage: string): u
  mod(a: uint256, b: uint256): uint256
  mod(a: uint256, b: uint256, errorMessage: string):
```

```
<<Interface>>
IUniswapV2Router01

External:
  factory(): address
  WETH(): address
  addLiquidity(tokenA: address, tokenB: address, amountADesired: uint, amountBDesired: uint, amountAMin: uint, amountBMin: uint, to: address, deadline: uint): (amountA: uint, amountB: uint, liquidity: uint)
  addLiquidityETH(token: address, amountTokenDesired: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint): (amountToken: uint, amountETH: uint, liquidity: uint)
  removeLiquidity(tokenA: address, tokenB: address, liquidity: uint, amountAMin: uint, amountBMin: uint, to: address, deadline: uint): (amountA: uint, amountB: uint)
  removeLiquidityETH(token: address, liquidity: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint): (amountToken: uint, amountETH: uint)
  removeLiquidityWithPermit(tokenA: address, tokenB: address, liquidity: uint, amountAMin: uint, amountBMin: uint, to: address, deadline: uint, approveMax: bool, v: uint8, r: bytes32, s: bytes32): (amountA: uint, amountB: uint)
  removeLiquidityETHWithPermit(token: address, liquidity: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint, approveMax: bool, v: uint8, r: bytes32, s: bytes32): (amountToken: uint, amountETH: uint)
  swapExactTokensForTokens(amountIn: uint, amountOutMin: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
  swapTokensForExactTokens(amountOut: uint, amountInMax: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
  swapExactETHForTokens(amountOutMin: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
  swapTokensForExactETH(amountOut: uint, amountInMax: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
  swapExactTokensForETH(amountIn: uint, amountOutMin: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
  swapETHForExactTokens(amountOut: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
  quote(amountA: uint, reserveA: uint, reserveB: uint): (amountB: uint)
  getAmountOut(amountIn: uint, reserveIn: uint, reserveOut: uint): (amountOut: uint)
  getAmountIn(amountOut: uint, reserveIn: uint, reserveOut: uint): (amountIn: uint)
  getAmountsOut(amountIn: uint, path: address[]): (amounts: uint[])
  getAmountsIn(amountOut: uint, path: address[]): (amounts: uint[])
```

```
ERC20

Private:
  _balances: mapping(address=>uint256)
  _allowances: mapping(address=>mapping(address=>uint256))
  _totalSupply: uint256
  _name: string
  _symbol: string
  _decimals: uint8

Internal:
  _transfer(sender: address, recipient: address, amount: uint256)
  _createTSupply(account: address, amount: uint256)
  _approve(owner: address, spender: address, amount: uint256)
  _beforeTokenTransfer(from: address, to: address, amount: uint256)
Public:
  constructor(name_: string, symbol_: string)
  name(): string
  symbol(): string
  decimals(): uint8
  totalSupply(): uint256
  balanceOf(account: address): uint256
  transfer(recipient: address, amount: uint256): bool
  allowance(owner: address, spender: address): uint256
  approve(spender: address, amount: uint256): bool
  transferFrom(sender: address, recipient: address, amount: uint256): bool
  increaseAllowance(spender: address, addedValue: uint256): bool
  decreaseAllowance(spender: address, subtractedValue: uint256): bool
```

```
<<Abstract>>
Ownable

Private:
  _owner: address

Public:
  <<event>> OwnershipTransferred(previousOwner: address, newOw
  <<modifier>> onlyOwner()
  constructor()
  owner(): address
  renounceOwnership()
  transferOwnership(newOwner: address)
```

```
<<Interface>>
IUniswapV2Router02

External:
  removeLiquidityETHSupportingFeeOnTransferTokens(token: address, liquidity: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint): (amountETH: uint)
  removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(token: address, liquidity: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint, approveMax: bool, v: uint8, r: byte
  swapExactTokensForTokensSupportingFeeOnTransferTokens(amountIn: uint, amountOutMin: uint, path: address[], to: address, deadline: uint)
  swapExactETHForTokensSupportingFeeOnTransferTokens(amountOutMin: uint, path: address[], to: address, deadline: uint)
  swapExactTokensForETHSupportingFeeOnTransferTokens(amountIn: uint, amountOutMin: uint, path: address[], to: address, deadline: uint)
```

```
<<Interface>>
IUniswapV2Factory

External:
  feeTo(): address
  feeToSetter(): address
  getPair(tokenA: address, tokenB: address): (pair: address)
  allPairs(uint): (pair: address)
  allPairsLength(): uint
  createPair(tokenA: address, tokenB: address): (pair: address)
  setFeeTo(address)
  setFeeToSetter(address)
Public:
  <<event>> PairCreated(token0: address, token1: address, pa
```

```
NANXA

Private:
  inSwap: bool
  _isExcludedFromFees: mapping(address=>bool)
Public:
  swapEnabled: bool
  uniswapV2Router: IUniswapV2Router02
  uniswapV2Pair: address
  marketingFee: uint256
  marketingWallet: address
  maxTransactionAmount: uint256
  swapTokensAtAmount: uint256

Private:
  swapTokensForBNB(tokenAmount: uint256, _to: address)
Internal:
  _transfer(from: address, to: address, amount: uint256)
External:
  setSwapTokensAtAmount(newAmt: uint256)
  setMarketingWallet(_newAddress: address)
Public:
  <<event>> ExcludeFromFees(account: address, isExcluded: bool)
  <<event>> SwapEnabledUpdated(enabled: bool)
  <<modifier>> lockTheSwap()
  constructor()
  excludeFromFees(account: address, excluded: bool)
  isExcludedFromFees(account: address): bool
  setSwapEnabled(_enabled: bool)
  setFee(_newFee: uint256)
  setMaxTransaction(_maxTxAmount: uint256)
```

Functions signature

```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
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)
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)
aaf467db  =>  _createTSupply(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
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,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)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
d73c29dc  =>  swapTokensForBNB(uint256,address)
c0246668  =>  excludeFromFees(address,bool)
4fbee193  =>  isExcludedFromFees(address)
afa4f3b2  =>  setSwapTokensAtAmount(uint256)
e01af92c  =>  setSwapEnabled(bool)
5d098b38  =>  setMarketingWallet(address)
69fe0e2d  =>  setFee(uint256)
ab5a1887  =>  setMaxTransaction(uint256)
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/NANXA.sol |
95e657909750c43e1b888f969da2f97480b06d95 |

 Contracts Description Table

|  Contract  |         Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
-------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | owner | Public ❗ |   |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
| **SafeMath** | Library |  |||
| └ | add | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | mul | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
| **ERC20** | Implementation | Context, IERC20 |||
| └ | <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 🔒 | 🛑  | |
| └ | _createTSupply | Internal 🔒 | 🛑  | |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑  | |
| **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❗️ |
| **NANXA** | Implementation | ERC20, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑  | ERC20 |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | swapTokensForBNB | Private 🔐 | 🛑  | lockTheSwap |
| └ | excludeFromFees | Public ❗️ | 🛑  | onlyOwner |
| └ | isExcludedFromFees | Public ❗️ |   |NO❗️ |
| └ | setSwapTokensAtAmount | External ❗️ | 🛑  | onlyOwner |
| └ | setSwapEnabled | Public ❗️ | 🛑  | onlyOwner |
| └ | setMarketingWallet | External ❗️ | 🛑  | onlyOwner |
| └ | setFee | Public ❗️ | 🛑  | onlyOwner |
| └ | setMaxTransaction | 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.

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.
✓ No 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.