Smart Contract
Security Audit
V1

Nexus Token Smart Contract

8/3/2022

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN

–

Table of Contents

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Platform**: Avalanche C-Chain
- **Contract Address**: 0x8d8A7Aa88b51DcC71722a0E7300cC798378a9573
- **Code Source:**

https://github.com/NexusDAODeFi/nexus-contracts

Token Information

- Name: NXS
- Total Supply: 1,000,000
- Holders:
- Total transactions:

Contracts address deployed to test net (AVAX)

Nexus Token Smart contract on AVAX test net to test write functions by the auditor.

https://testnet.snowtrace.io/address/0x8d8a7aa88b51dcc71722a0e7300cc798378a9573

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**.Because the team fix the high issue.

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

The files:

NexusERC20.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
NexusERC20.sol	**d2b1aea93e637a61c76766674683df5f7ad5c2ab88cd6eb1fd7582debe6887e0**	0x8d8A7Aa88b51DcC71722a0E7300cC798378a9573

- Contract: NexusERC20
- 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**
useAntibot	✓	Read / public	**Passed**
totalBurned	x	Read / public	**Not Passed**
taxReceiver	✓	Read / public	**Passed**
taxPercent	✓	Read / public	**Passed**
taxEnabled	✓	Read / public	**Passed**

NEXUS	✓	Read / public	**Passed**
pair	✓	Read / public	**Passed**
router	✓	Read / public	**Passed**
isAuthorized	✓	Read / public	**Passed**
isTaxFree	✓	Read / public	**Passed**
isBlackListed	✓	Read / public	**Passed**
isInBlacklist	✓	Read / public	**Passed**
burnPercent	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
updateTaxExemption	✓	Write / public	**Passed**
updateBlacklist	✓	Write / public	**Passed**
updateAuthorization	✓	Write / public	**Passed**
toggleUseAntibot	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
toggleTaxEnable	✓	Write / public	**Passed**
setTaxReceiver	✓	Write / public	**Passed**
setTaxPercent	✓	Write / public	**Passed**
burn	✓	Write / public	**Passed**
setBurnPercent	✓	Write / public	**Passed**
setRouterAddress	✓	Write / public	**Passed**
setPairAddress	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
setNEXUEAddress	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Design Logic.	**Passed**
6	Timestamp dependence.	**Passed**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed with notes**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**Passed**

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical:

#The owner or any authorized address can burn the user's funds

Description

The owner or any authorized address has the ability to burn as much as he wants; this represents a risk for the users because in that case their funds will be burned without his/her permission in the smart contract.

```
function burn(address from, uint256 amount) external onlyAuthorized {
        _burn(from, amount);
        emit TokensBurned(amount, from, _msgSender());
    }
```

Remediation

There are two possibilities to remediate the risk. Make this function internal which no one can control it. The second one is making burn function burn from the total supply not from user's funds which is recommended by the auditor.

Status: Closed. Fixed in version2.

High:

No High severity vulnerabilities were found

Medium:

No Medium severity vulnerabilities were found.

Low:

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.9 instead of ^0.8.9). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

Remediation

Remove the ^ sign to lock the pragma version.

Status: Acknowledged.

#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

Owner can change Burn and Tax Fees or make it = zero.

Owner can add / remove any address to Blacklist.

```
function updateBlacklist(address account, bool value) external onlyOwner {
        isBlacklisted[account] = value;
        emit BlacklistUpdated(account, value);
    }
    function toggleTaxEnabled() external onlyOwner {
        taxEnabled = !taxEnabled;
    }
    function setTaxPercent(uint256 percent) external onlyOwner {
        taxPercent = percent;
    }

    function setBurnPercent(uint256 percent) external onlyOwner {
        burnPercent = percent;
    }
```

Remediation

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

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

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#totalBurned function doesn’t work

Description
The totalBurned read function doesn’t work it keep sending zero burns from the total supply after burning, it should be able to read the total burn from the deployment moment till it is called.

Status: Acknowledged

Constant calculations in the contract

Description
recalculated initialization will save 2847 units of gas in deployment

```
_mint(_msgSender(), 1_000_000 * 10**decimals());
```

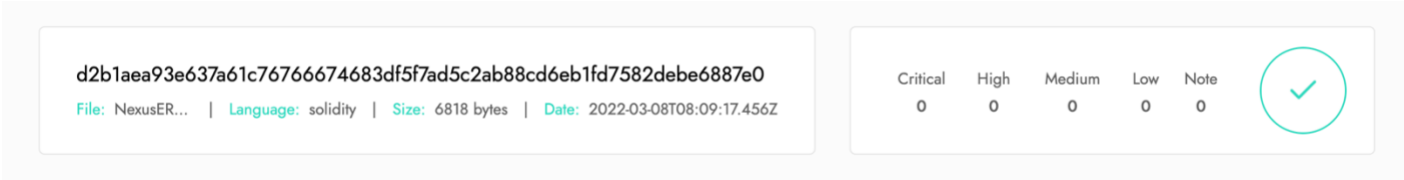
Recommendation
Replace the initialization as

```
_mint(_msgSender(), 1000000000000000000000000);
```

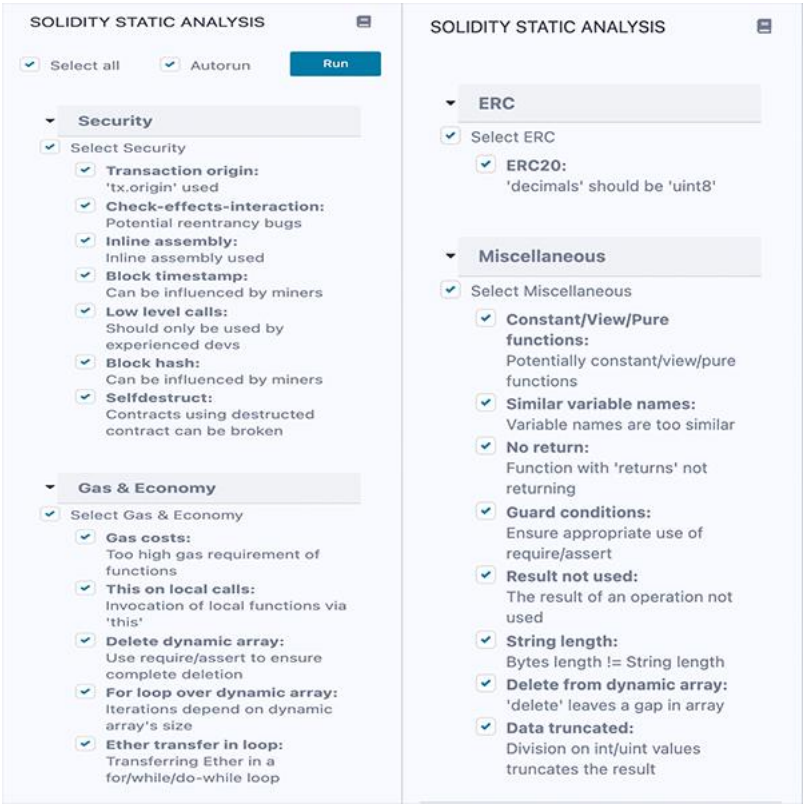
Status: Acknowledged

Automatic Testing

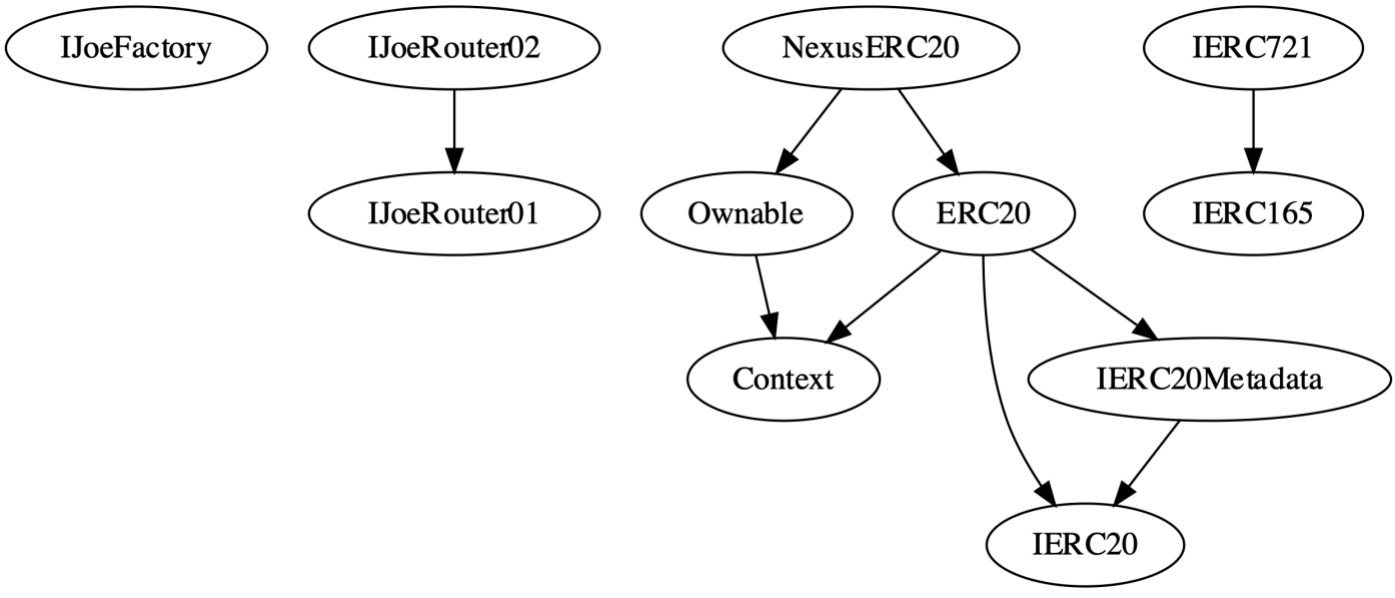
1- Check for security



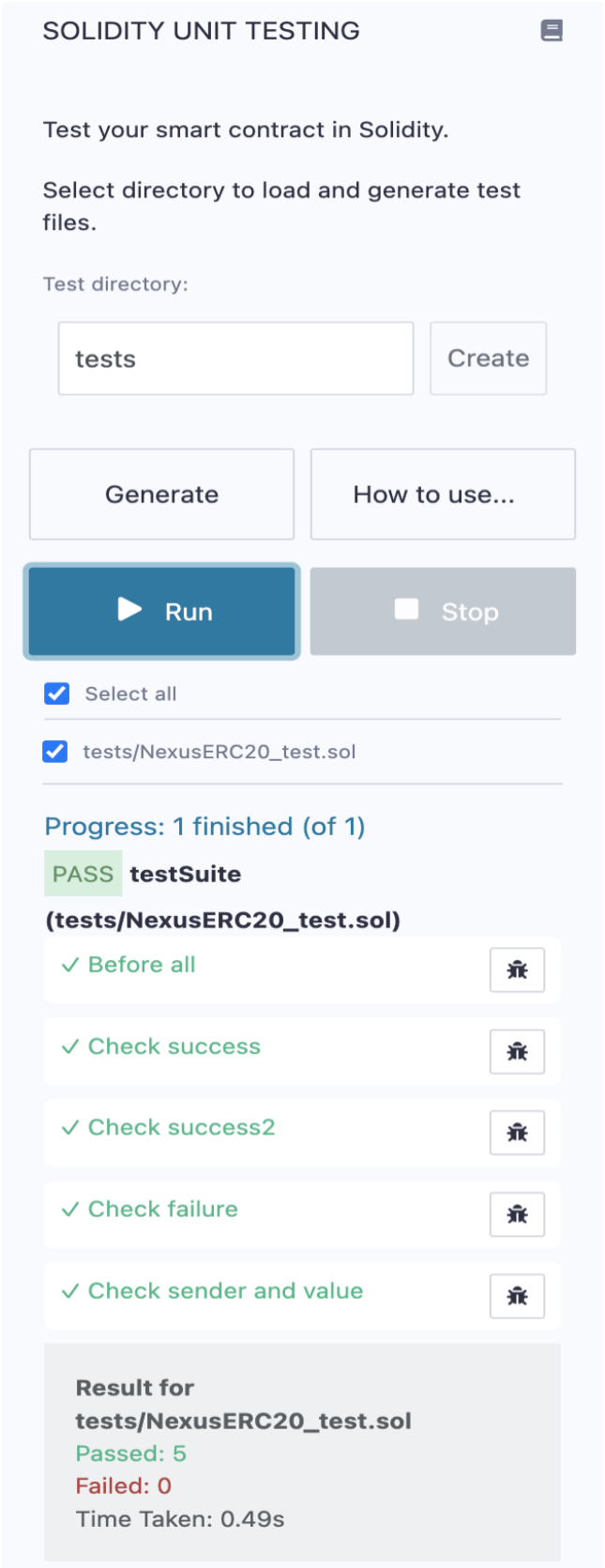
2- SOLIDITY STATIC ANALYSIS



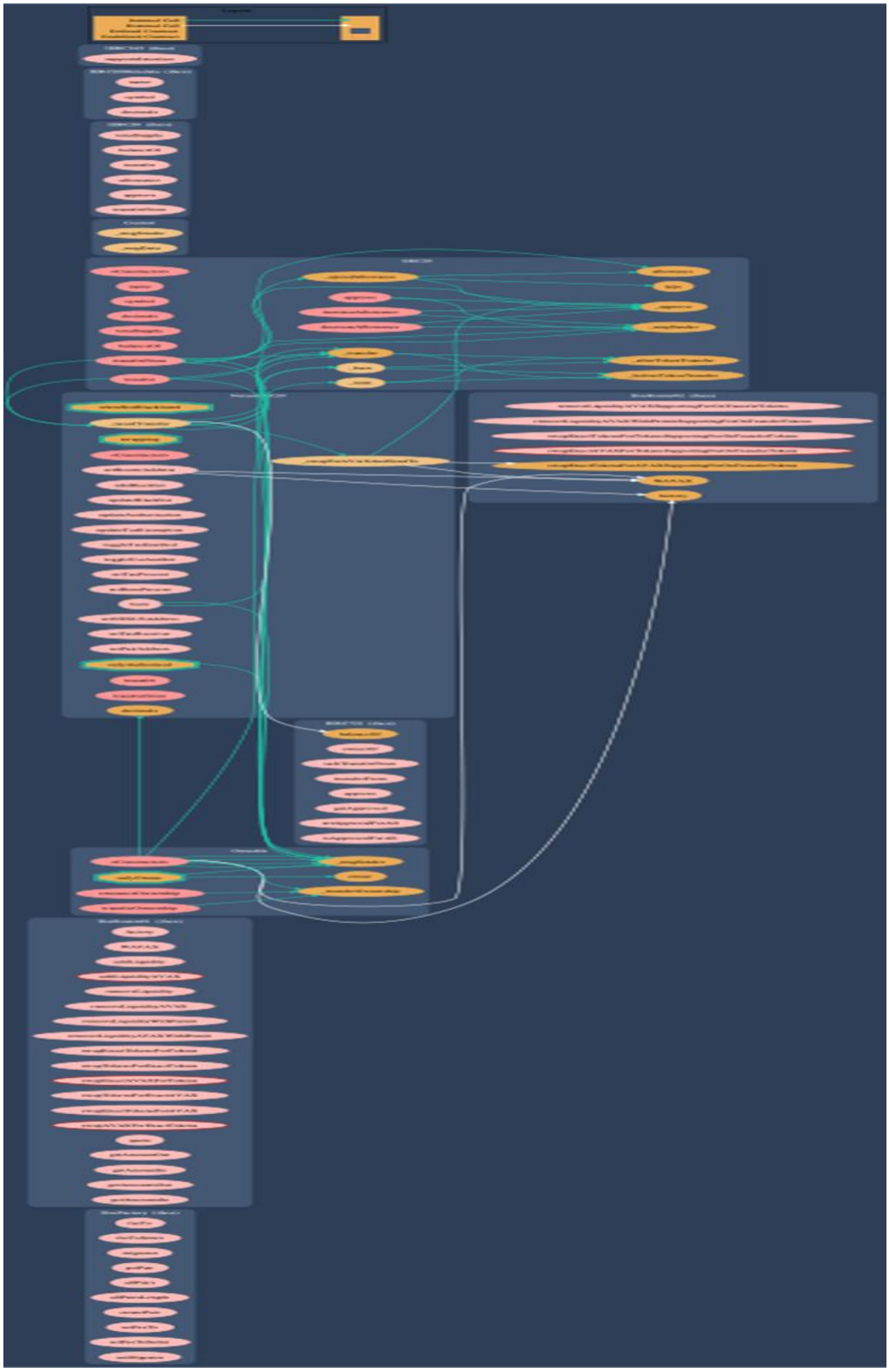
3- Inheritance graph



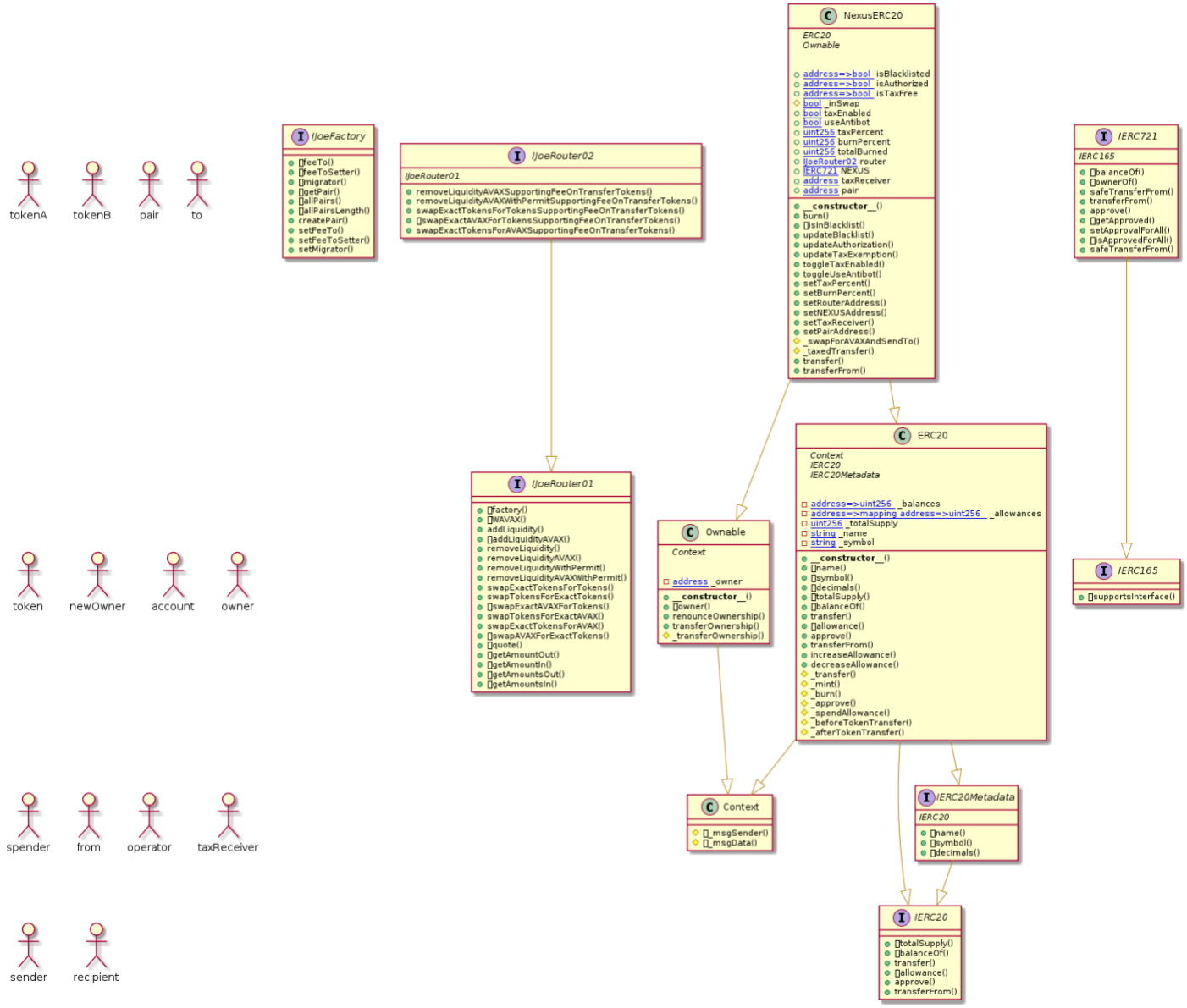
4- SOLIDITY UNIT TESTING



5- Call graph



Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
61178386  =>  setTaxPercent(uint256)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
7cd07e47  =>  migrator()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
23cf3118  =>  setMigrator(address)
c45a0155  =>  factory()
73b295c2  =>  WAVAX()
e8e33700  =>
addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f91b3f72  =>  addLiquidityAVAX(address,uint256,uint256,uint256,address,uint256)
baa2abde  =>
removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
33c6b725  =>  removeLiquidityAVAX(address,uint256,uint256,uint256,address,uint256)
2195995c  =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,b
ool,uint8,bytes32,bytes32)
2c407024  =>
removeLiquidityAVAXWithPermit(address,uint256,uint256,uint256,address,uint256,bool,
uint8,bytes32,bytes32)
38ed1739  =>  swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee  =>  swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
a2a1623d  =>  swapExactAVAXForTokens(uint256,address[],address,uint256)
7a42416a  =>  swapTokensForExactAVAX(uint256,uint256,address[],address,uint256)
676528d1  =>  swapExactTokensForAVAX(uint256,uint256,address[],address,uint256)
8a657e67  =>  swapAVAXForExactTokens(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[])
73bc79cf  =>
removeLiquidityAVAXSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,ad
dress,uint256)
9fc27226  =>
removeLiquidityAVAXWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,
uint256,address,uint256,bool,uint8,bytes32,bytes32)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],add
ress,uint256)
c57559dd  =>
swapExactAVAXForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint2
56)
762b1562  =>
swapExactTokensForAVAXSupportingFeeOnTransferTokens(uint256,uint256,address[],addre
ss,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
```

```
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
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)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
1532335e  =>  _spendAllowance(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
8f811a1c  =>  _afterTokenTransfer(address,address,uint256)
01ffc9a7  =>  supportsInterface(bytes4)
6352211e  =>  ownerOf(uint256)
42842e0e  =>  safeTransferFrom(address,address,uint256)
081812fc  =>  getApproved(uint256)
a22cb465  =>  setApprovalForAll(address,bool)
e985e9c5  =>  isApprovedForAll(address,address)
b88d4fde  =>  safeTransferFrom(address,address,uint256,bytes)
9dc29fac  =>  burn(address,uint256)
9caf9b00  =>  isInBlacklist(address)
9155e083  =>  updateBlacklist(address,bool)
ba2c4afc  =>  updateAuthorization(address,bool)
e3de2c1c  =>  updateTaxExemption(address,bool)
9c648e44  =>  toggleTaxEnabled()
47a91dfc  =>  toggleUseAntibot()
bb1570da  =>  setBurnPercent(uint256)
41cb87fc  =>  setRouterAddress(address)
5256af73  =>  setNEXUSAddress(address)
cd8de42c  =>  setTaxReceiver(address)
a22d4832  =>  setPairAddress(address)
4a14afe2  =>  _swapForAVAXAndSendTo(address,uint256)
7e12ada3  =>  _taxedTransfer(address,address,uint256)
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/NexusERC20.sol	8ec964c30c02150611bd4245780e80760ec85fac

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IJoeFactory	Interface			
└	feeTo	External ❗	NO❗	
└	feeToSetter	External ❗	NO❗	
└	migrator	External ❗	NO❗	
└	getPair	External ❗	NO❗	
└	allPairs	External ❗	NO❗	
└	allPairsLength	External ❗	NO❗	
└	createPair	External ❗	🛑	NO❗
└	setFeeTo	External ❗	🛑	NO❗
└	setFeeToSetter	External ❗	🛑	NO❗
└	setMigrator	External ❗	🛑	NO❗
IJoeRouter01	Interface			
└	factory	External ❗	NO❗	
└	WAVAX	External ❗	NO❗	
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityAVAX	External ❗	💵	NO❗
└	removeLiquidity	External ❗	🛑	NO❗
└	removeLiquidityAVAX	External ❗	🛑	NO❗
└	removeLiquidityWithPermit	External ❗	🛑	NO❗
└	removeLiquidityAVAXWithPermit	External ❗	🛑	NO❗
└	swapExactTokensForTokens	External ❗	🛑	NO❗
└	swapTokensForExactTokens	External ❗	🛑	NO❗
└	swapExactAVAXForTokens	External ❗	💵	NO❗
└	swapTokensForExactAVAX	External ❗	🛑	NO❗
└	swapExactTokensForAVAX	External ❗	🛑	NO❗
└	swapAVAXForExactTokens	External ❗	💵	NO❗
└	quote	External ❗	NO❗	
└	getAmountOut	External ❗	NO❗	
└	getAmountIn	External ❗	NO❗	
└	getAmountsOut	External ❗	NO❗	
└	getAmountsIn	External ❗	NO❗	
IJoeRouter02	Interface	IJoeRouter01		
└	removeLiquidityAVAXSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityAVAXWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗

└	swapExactAVAXForTokensSupportingFeeOnTransferTokens	External ❗	💰	NO❗
└	swapExactTokensForAVAXSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
IERC165	Interface			
└	supportsInterface	External ❗		NO❗
IERC721	Interface	IERC165		
└	balanceOf	External ❗		NO❗
└	ownerOf	External ❗		NO❗
└	safeTransferFrom	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	approve	External ❗	🛑	NO❗
└	getApproved	External ❗		NO❗

└	setApprovalForAll	External ❗	🛑	NO❗
└	isApprovedForAll	External ❗		NO❗
└	safeTransferFrom	External ❗	🛑	NO❗
NexusERC20	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	burn	External ❗	🛑	onlyAuthorized
└	isInBlacklist	External ❗		NO❗
└	updateBlacklist	External ❗	🛑	onlyOwner
└	updateAuthorization	External ❗	🛑	onlyOwner
└	updateTaxExemption	External ❗	🛑	onlyOwner
└	toggleTaxEnabled	External ❗	🛑	onlyOwner
└	toggleUseAntibot	External ❗	🛑	onlyOwner
└	setTaxPercent	External ❗	🛑	onlyOwner
└	setBurnPercent	External ❗	🛑	onlyOwner
└	setRouterAddress	External ❗	🛑	onlyOwner
└	setNEXUSAddress	External ❗	🛑	onlyOwner
└	setTaxReceiver	External ❗	🛑	onlyOwner
└	setPairAddress	External ❗	🛑	onlyOwner
└	_swapForAVAXAndSendTo	Internal 🔒	🛑	swapping
└	_taxedTransfer	Internal 🔒	🛑	whenNotBlacklisted whenNotBlacklisted whenNotBlacklisted
└	transfer	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗

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 volatile code.
✓ Not many high severity issues were found.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.