Smart Contract
Security Audit
V1

Nexus Vault Smart Contract

9/3/2022

https://saferico.com/

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

–

Table of Contents

Table of Contents

Background

Project Information

- Smart Contract Information
- Executive Summary

File and Function Level Report

- **File in Scope:**

Issues Checking Status

- Severity Definitions
- Audit Findings

Automatic testing

- Testing proves
- Inheritance graph
- Call graph

Unified Modeling Language (UML)

Functions signature

Automatic general report

Conclusion

Disclaimer

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**: 0x9105F8d7E28c1fEC6fbe547fd2825c816CFd10c7

- **Code Source:**

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

- **Contract Type: Stake Contract for Nexus Ecosystem.**

Contracts address deployed to test net (AVAX)

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

https://testnet.snowtrace.io/address/0x9105f8d7e28c1fec6fbe547fd2825c816cfd10c7

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

The files:

NexusVault.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
NexusVault.sol	**689efa92d90a557e4f6beb51655c9dfb715198929d91610d2af744aa41c03d0c**	0x9105F8d7E28c1fEC6fbe547fd2825c816CFd10c7

- Contract: NexusVault
- Inherit: ERC721Holder, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
getStakerOf	✓	Read / public	**Passed**
getAllRewards	✓	Read / public	**Passed**
getRewardsByTokenId	✓	Read / public	**Passed**
getTokensStaked	✓	Read / public	**Passed**
isWalletLimited	✓	Read / public	**Passed**
maxPerWallet	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
NEXUS	✓	Read / public	**Passed**
tokenIdToTimestamp	✓	Read / public	**Passed**
tokenIdToStaker	✓	Read / public	**Passed**
stakerToTokenIds	✓	Read / public	**Passed**
NXS	✓	Read / public	**Passed**

renounceOwnership	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
stake	✓	Write / public	**Passed**
unStake	✓	Write / public	**Passed**
claim	✓	Write / public	**Passed**
setNXSAddress	✓	Write / public	**Passed**
setNEXUSAddress	✓	Write / public	**Passed**
setMaxPerWallet	✓	Write / public	**Passed**
setIsWalletLimited	✓	Write / public	**Passed**
onERC721Received	✓	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:

No Critical severity vulnerabilities were found.

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 the max per wallet.

Owner can add / remove the wallet limits statement.

```
function setIsWalletLimited(bool value) external onlyOwner {
        isWalletLimited = value;
    }

    function setMaxPerWallet(uint256 limit) external onlyOwner {
        maxPerWallet = limit;
    }
```

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

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Unused Address library

Description
The main contract inherits: ERC721Holder, Ownable. And else import address library but without using it.

Remediation
Remove address library for the main contract save some gas fees.

Status: Acknowledged

Automatic Testing

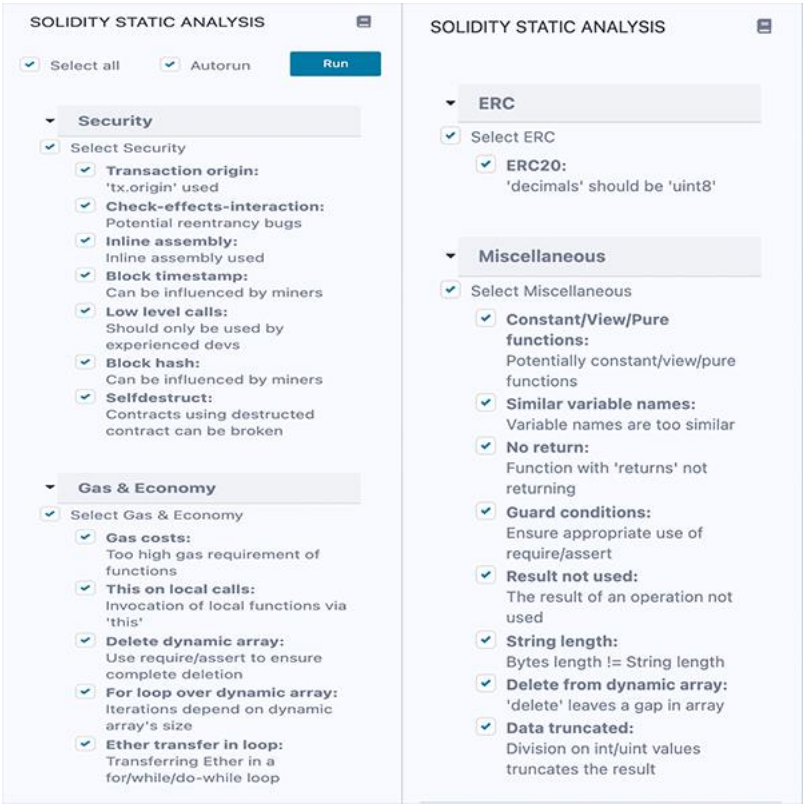
1- Check for security

689efa92d90a557e4f6beb51655c9dfb715198929d91610d2af744aa41c03d0c

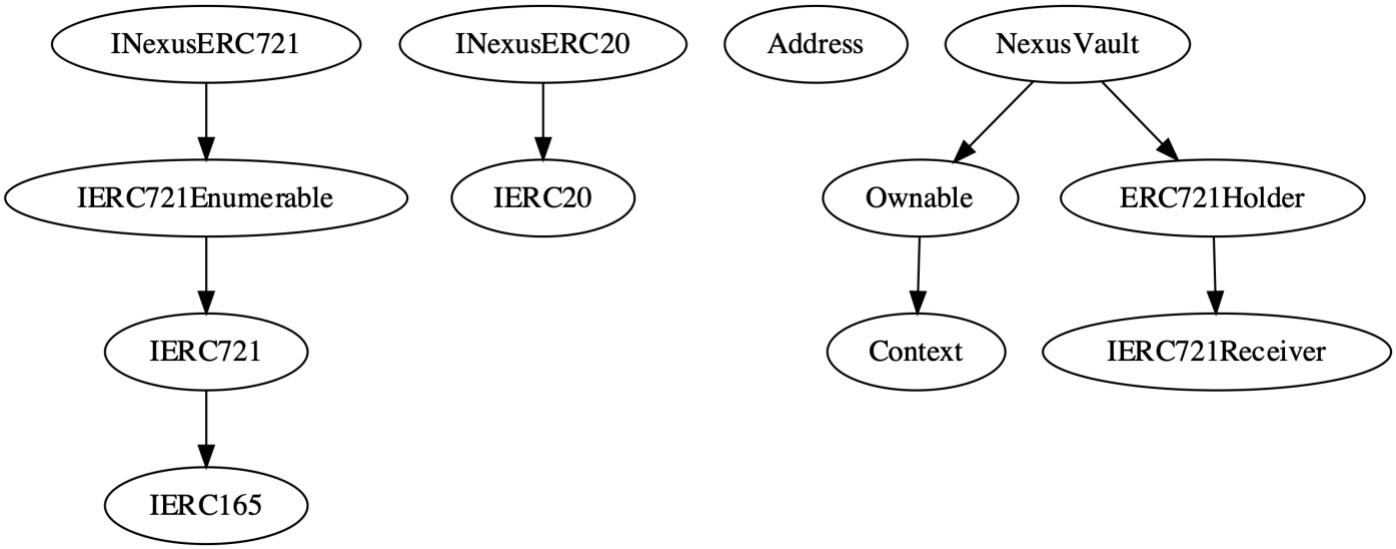
File: NexusVa... | Language: solidity | Size: 5918 bytes | Date: 2022-03-09T10:09:18.675Z

Critical	High	Medium	Low	Note
0	0	0	0	0

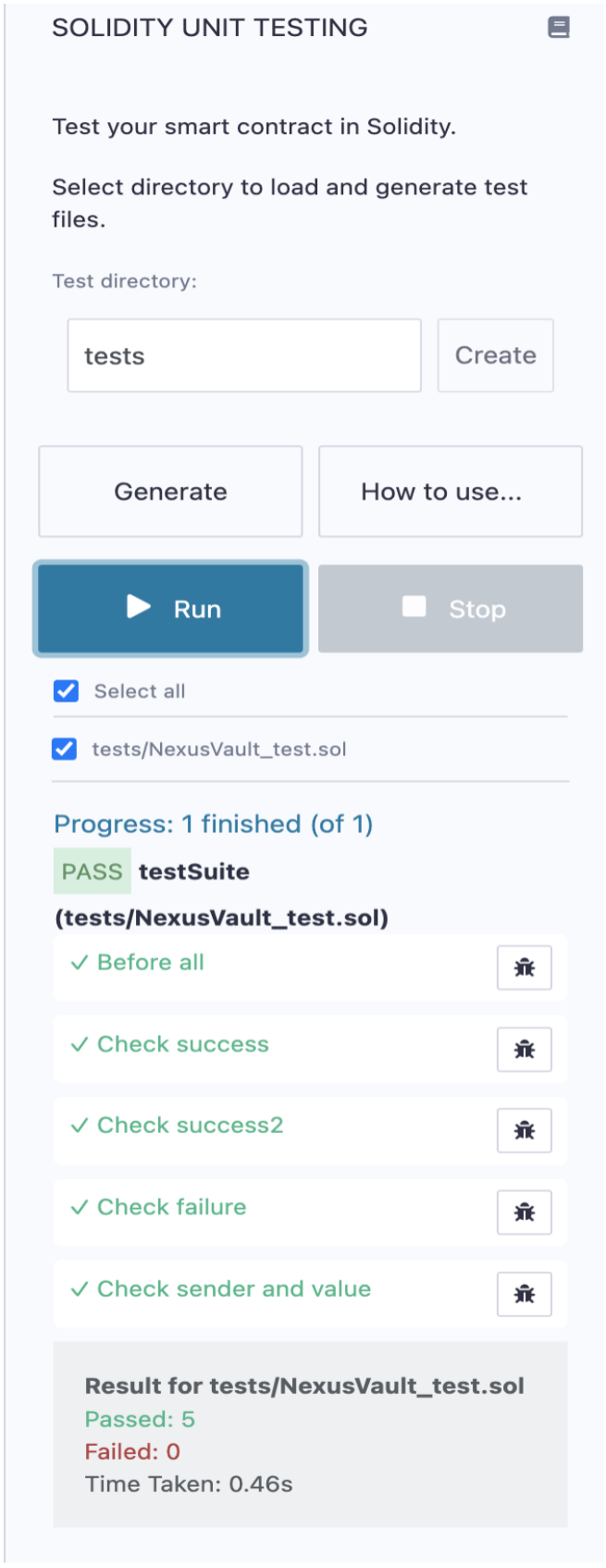
2- SOLIDITY STATIC ANALYSIS



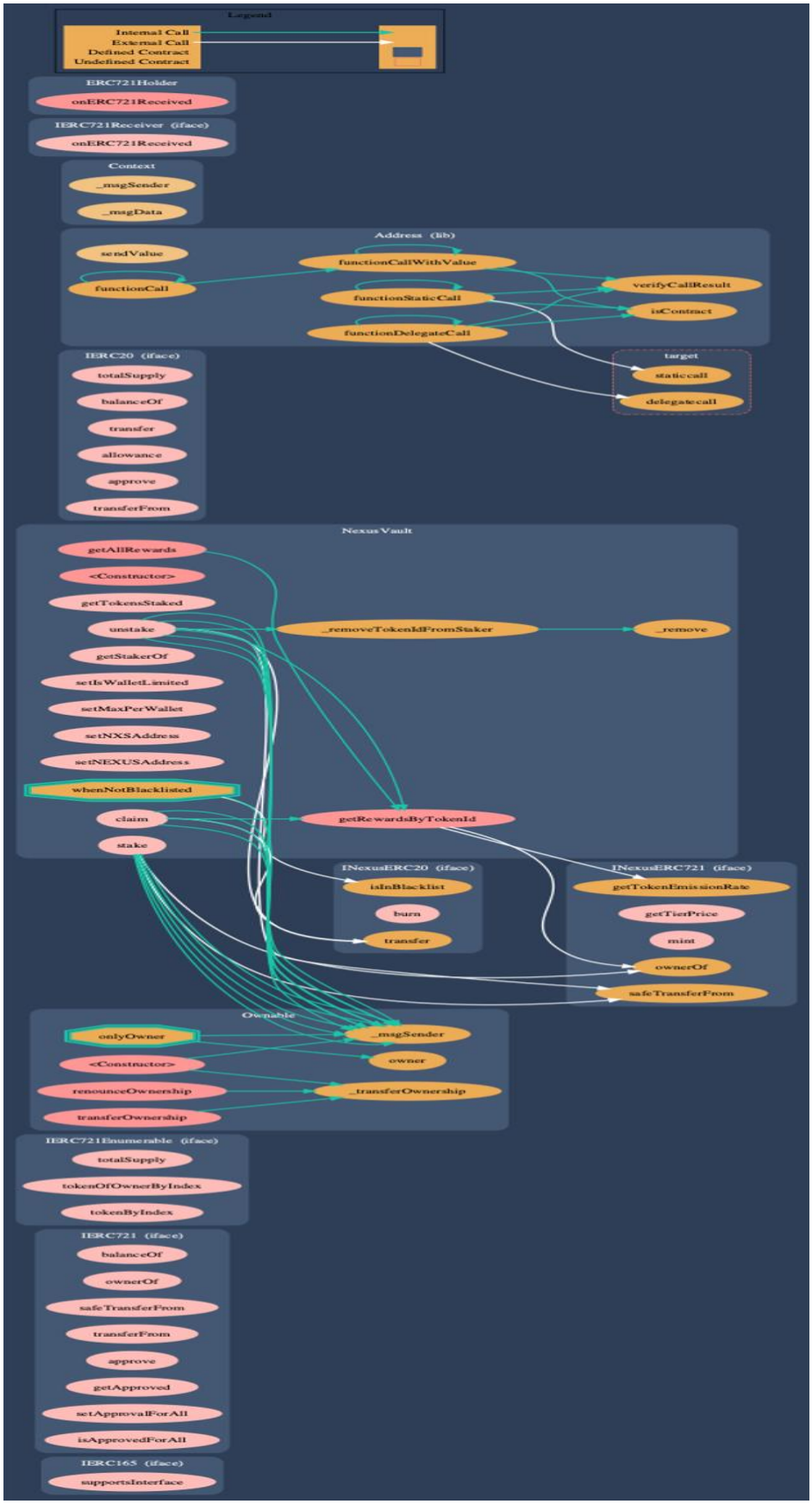
3- Inheritance graph



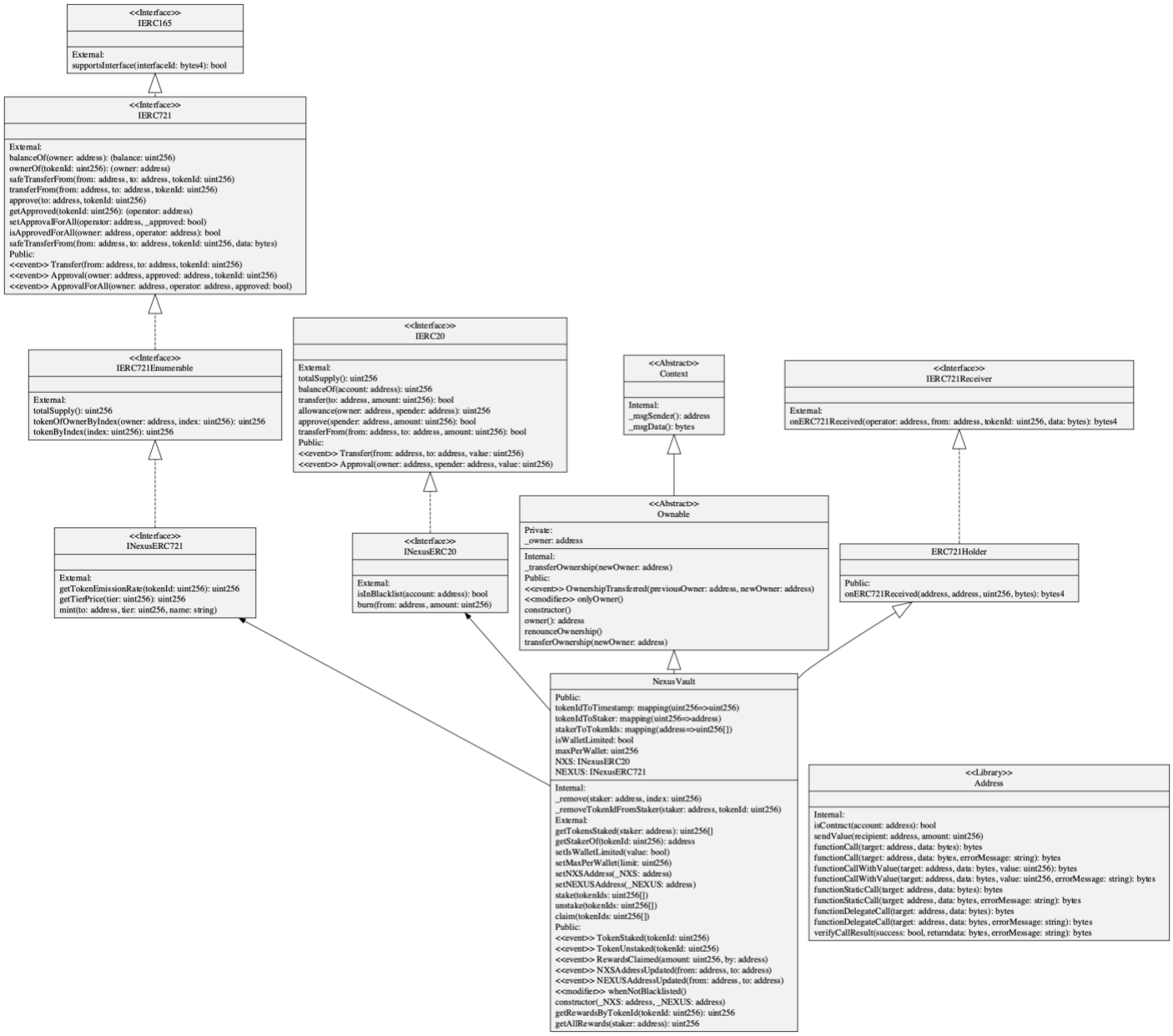
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ERC721Holder
onERC721Received
IERC721Receiver (iface)
onERC721Received
Context
_msgSender
_msgData
Address (lib)
sendValue
functionCallWithValue
functionCall
functionStaticCall
functionDelegateCall
verifyCallResult
isContract
target
staticcall
delegatecall
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
NexusVault
getAllRewards
<Constructor>
getTokensStaked
unstake
_removeTokenIdFromStaker
_remove
getStakerOf
setIsWalletLimited
setMaxPerWallet
setNXSAddress
setNEXUSAddress
whenNotBlacklisted
claim
stake
getRewardsByTokenId
INexusERC20 (iface)
isInBlacklist
burn
transfer
INexusERC721 (iface)
getTokenEmissionRate
getTierPrice
mint
ownerOf
safeTransferFrom
Ownable
onlyOwner
<Constructor>
renounceOwnership
transferOwnership
_msgSender
owner
_transferOwnership
IERC721Enumerable (iface)
totalSupply
tokenOfOwnerByIndex
tokenByIndex
IERC721 (iface)
balanceOf
ownerOf
safeTransferFrom
transferFrom
approve
getApproved
setApprovalForAll
isApprovedForAll
IERC165 (iface)
supportsInterface

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
94953219  =>  _removeTokenIdFromStaker(address,uint256)
01ffc9a7  =>  supportsInterface(bytes4)
70a08231  =>  balanceOf(address)
6352211e  =>  ownerOf(uint256)
42842e0e  =>  safeTransferFrom(address,address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
095ea7b3  =>  approve(address,uint256)
081812fc  =>  getApproved(uint256)
a22cb465  =>  setApprovalForAll(address,bool)
e985e9c5  =>  isApprovedForAll(address,address)
b88d4fde  =>  safeTransferFrom(address,address,uint256,bytes)
18160ddd  =>  totalSupply()
2f745c59  =>  tokenOfOwnerByIndex(address,uint256)
4f6ccce7  =>  tokenByIndex(uint256)
5658892b  =>  getTokenEmissionRate(uint256)
252a8875  =>  getTierPrice(uint256)
d3fc9864  =>  mint(address,uint256,string)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
9caf9b00  =>  isInBlacklist(address)
9dc29fac  =>  burn(address,uint256)
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
c21d36f3  =>  functionStaticCall(address,bytes)
dbc40fb9  =>  functionStaticCall(address,bytes,string)
ee33b7e2  =>  functionDelegateCall(address,bytes)
57387df0  =>  functionDelegateCall(address,bytes,string)
946b5793  =>  verifyCallResult(bool,bytes,string)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
150b7a02  =>  onERC721Received(address,address,uint256,bytes)
52eb7796  =>  getTokensStaked(address)
515ec105  =>  getRewardsByTokenId(uint256)
362a3fad  =>  getAllRewards(address)
8c6f90ba  =>  getStakerOf(uint256)
5b3f5b15  =>  setIsWalletLimited(bool)
e268e4d3  =>  setMaxPerWallet(uint256)
f2f48546  =>  setNXSAddress(address)
5256af73  =>  setNEXUSAddress(address)
0fbf0a93  =>  stake(uint256[])
e449f341  =>  unstake(uint256[])
6ba4c138  =>  claim(uint256[])
45a338cd  =>  _remove(address,uint256)
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/NexusVault.sol	49029b2c5d346266d290e3ed37cbd66adbbed910

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
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❗
IERC721Enumerable	Interface	IERC721		
└	totalSupply	External ❗		NO❗
└	tokenOfOwnerByIndex	External ❗		NO❗
└	tokenByIndex	External ❗		NO❗
INexusERC721	Interface	IERC721Enumerable		
└	getTokenEmissionRate	External ❗		NO❗
└	getTierPrice	External ❗		NO❗
└	mint	External ❗	🛑	NO❗
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
INexusERC20	Interface	IERC20		
└	isInBlacklist	External ❗		NO❗
└	burn	External ❗	🛑	NO❗
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	

└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	verifyCallResult	Internal 🔒		
Context	Implementation				
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
IERC721Receiver	Interface				
└	onERC721Received	External ❗	🛑	NO❗
ERC721Holder	Implementation	IERC721Receiver		
└	onERC721Received	Public ❗	🛑	NO❗
NexusVault	Implementation	ERC721Holder, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	getTokensStaked	External ❗		NO❗
└	getRewardsByTokenId	Public ❗		NO❗
└	getAllRewards	Public ❗		NO❗
└	getStakerOf	External ❗		NO❗
└	setIsWalletLimited	External ❗	🛑	onlyOwner
└	setMaxPerWallet	External ❗	🛑	onlyOwner
└	setNXSAddress	External ❗	🛑	onlyOwner
└	setNEXUSAddress	External ❗	🛑	onlyOwner
└	stake	External ❗	🛑	whenNotBlacklisted
└	unstake	External ❗	🛑	whenNotBlacklisted
└	claim	External ❗	🛑	whenNotBlacklisted
└	_remove	Internal 🔒	🛑	
└	_removeTokenIdFromStaker	Internal 🔒	🛑	

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.