NEXXO TOKEN SMART CONTRACT AUDIT

FOR NEXXO SG PTE. LTD.

18.06.2020

Made in Germany by Chainsulting.de

Smart Contract Audit - NEXXO Token

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of NEXXO SG PTE. LTD. . If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Previous Audit: https://github.com/chainsulting/Smart-Contract-Security-Audits/tree/master/Nexxo/2019

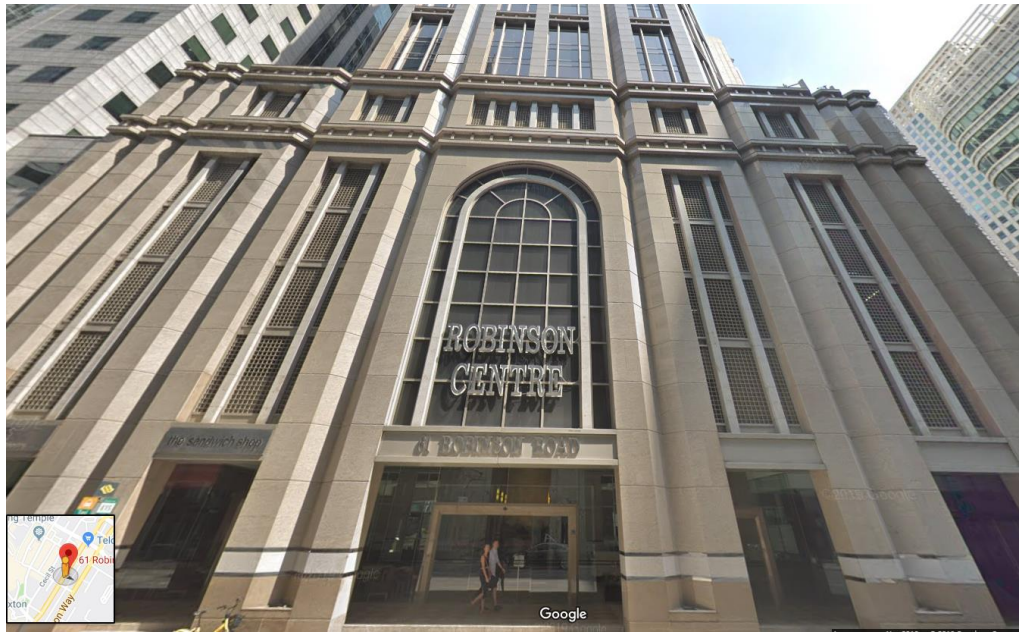
Major Versions / Date	Description	Author
0.1 (16.06.2020)	Layout	Y. Heinze
0.5 (18.06.2020)	Automated Security Testing Manual Security Testing	Y. Heinze
1.0 (19.06.2020)	Summary and Recommendation	Y. Heinze
1.1 (22.06.2020)	Adding of MythX	Y. Heinze

2. About the Project and Company

Company address:

NEXXO SG PTE. LTD.
61 ROBINSON ROAD #19-02
ROBINSON CENTRE
SINGAPORE 068893

CERTIFICATION OF INCORPORATION NO: 201832832R
LEGAL REPRESENTIVE: NEBIL BEN AISSA



2.1 Project Overview:

The world's first global blockchain-powered small business financial services platform. Nexxo is a multi-national company (currently incorporated in Qatar, UAE, India, Pakistan, Singapore and Cyprus Eurozone); it provides financial services to small businesses in the Middle East and emerging markets.

Nexxo financial services are bank accounts with an IBAN (International Bank Account Number), MasterCard powered Salary Cards, electronic commerce, Point of Sale, bill payment, invoicing as well as (in the future) loans and financing facilities. These solutions are offered using blockchain technology which reduces the cost of the service, as well as help small businesses grow their revenues, lower costs and achieve a better life for themselves and their families.

A Very unique characteristic of NEXXO is that it partners with locally licensed banks, and operates under approval of local central banks; its blockchain is architected to be in full compliance with local central banks, and its token is designed as a reward and discount token, thus not in conflict with locally regulated national currencies. All localized Nexxo Blockchains are Powered by IBM Hyperledger, and connected onto a multi-country international blockchain called NEXXONET.

NEXXO operates in multiple countries, it generates profits of Approximately $4.0 Mil USD (audited by Deloitte) and is managed by a highly skilled and experienced team.

Security Notice: Re-deploy of the Smart Contract due to a security breach on Digifinex platform.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts

1. SafeMath.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol
2. ERC20Burnable.sol
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/token/ERC20/ERC20Burnable.sol
3. ERC20.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol
4. IERC20.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/IERC20.sol
5. Ownable.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol
6. Pausable.sol
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/lifecycle/Pausable.sol
7. PauserRole.sol
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/access/roles/PauserRole.sol
8. Roles.sol
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/access/Roles.sol
9. SafeERC20.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/SafeERC20.sol
10. TokenVesting.sol
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/drafts/TokenVesting.sol

4.3 Tested Contract Files

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (SHA256)
NexxoToken.sol	a84775efa28bfa3a20644ddf60688d605d5e78d18a4b765950ccad8fcd05cc9a

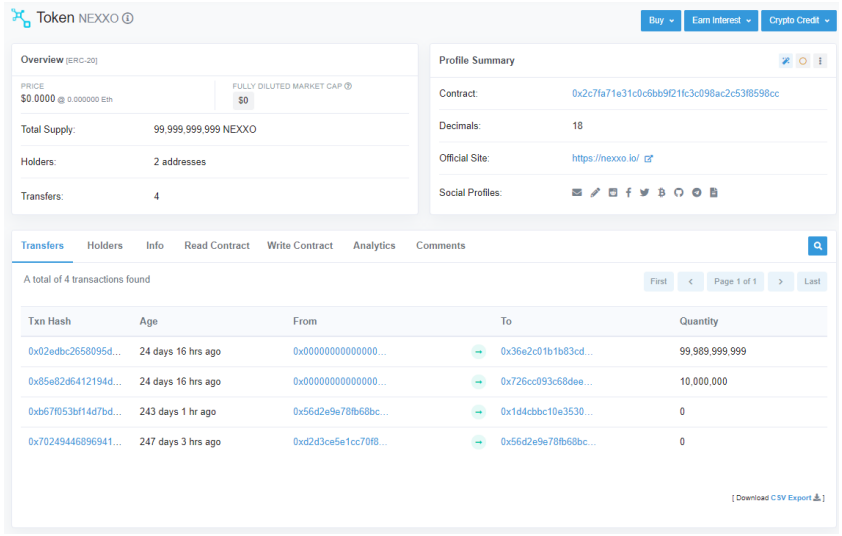
Source:
https://raw.githubusercontent.com/chainsulting/Smart-Contract-Security-Audits/master/Nexxo/2020/NexxoToken.sol

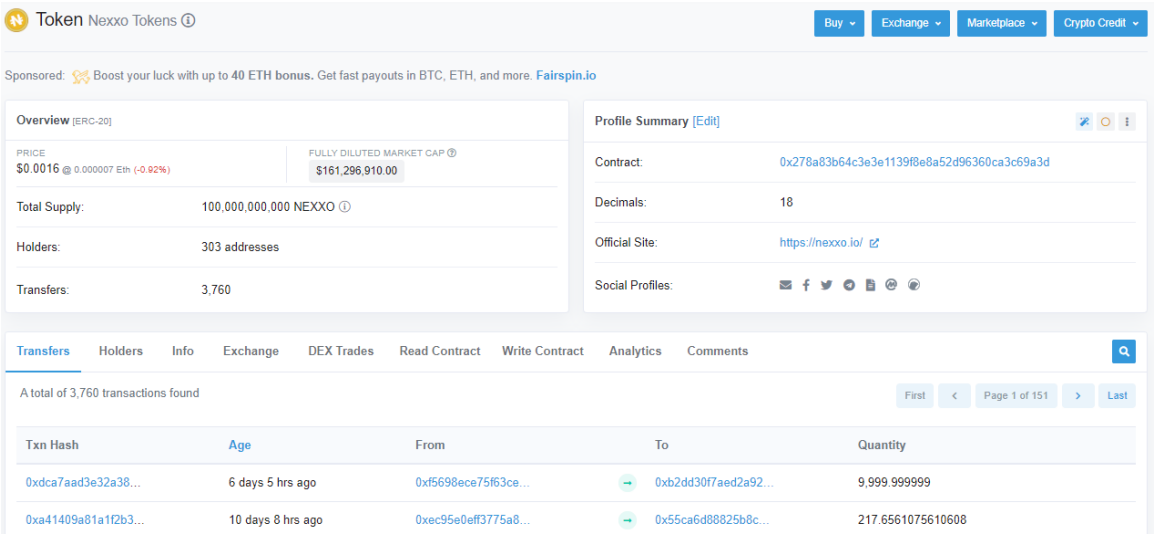
4.4 Contract Specifications

Language	Solidity
Token Standard	ERC20
Most Used Framework	OpenZeppelin
Compiler Version	0.5.3
Burn Function	Yes
Mint Function	No (Fixed total supply)
Lock Mechanism	Yes
Vesting Function	Yes
Ticker Symbol	NEXXO
Total Supply	100 000 000 000
Decimals	18

4.5 Special Security Note

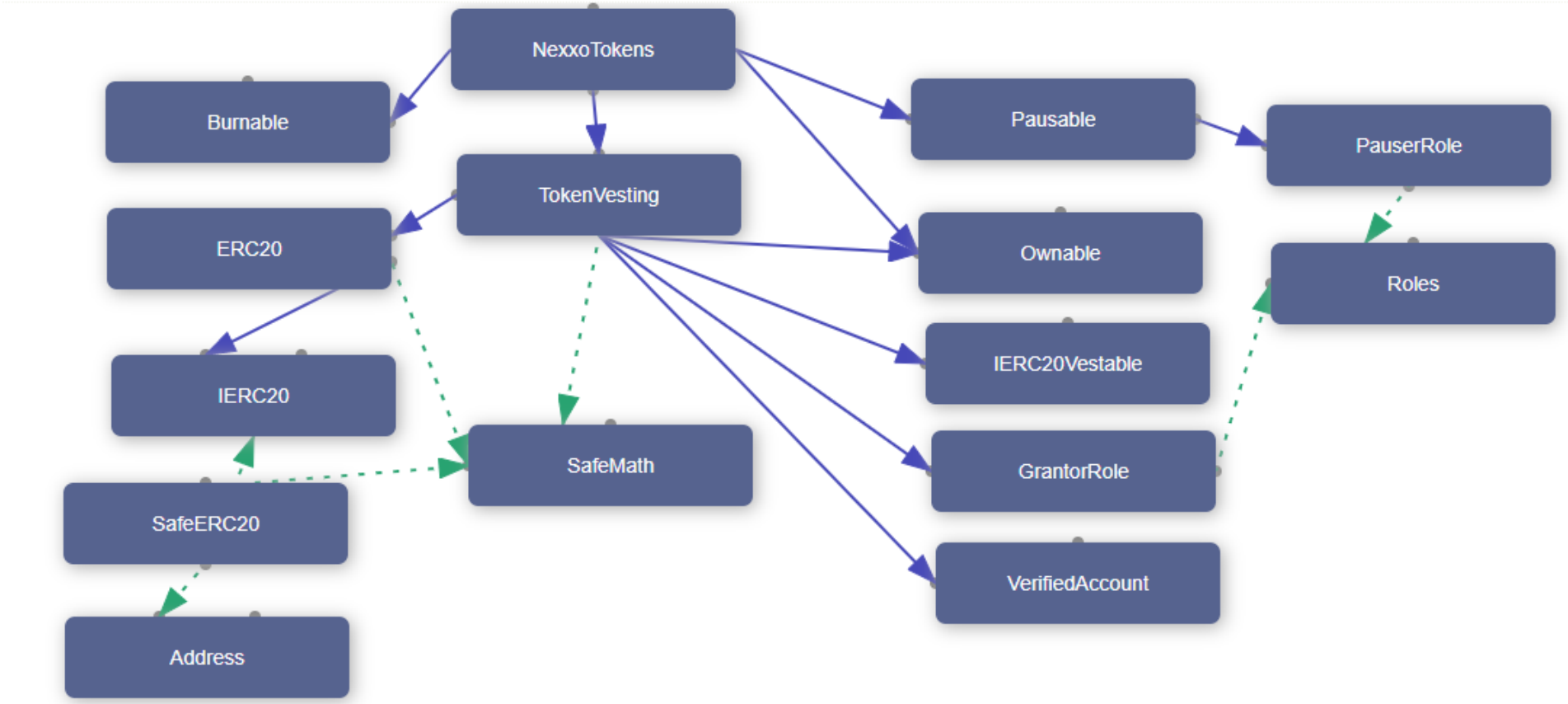
The following Smart Contracts are outdated and not anymore used by Nexxo Network Company. DON’T USE IT !
https://etherscan.io/token/0x2c7fa71e31c0c6bb9f21fc3c098ac2c53f8598cc
https://etherscan.io/token/0x278a83b64c3e3e1139f8e8a52d96360ca3c69a3d





5. Summary of Smart Contract

5.1 Visualized Dependencies



5.2 Functions

contract	func	visibility	modifiers	stateMutability
Address	isContract	internal		view
Address	toPayable	internal		pure
SafeMath	add	internal		pure
SafeMath	sub	internal		pure
SafeMath	mul	internal		pure
SafeMath	div	internal		pure
SafeMath	mod	internal		pure
SafeERC20	safeTransfer	internal		
SafeERC20	safeTransferFrom	internal		
SafeERC20	safeApprove	internal		
SafeERC20	safeIncreaseAllowance	internal		
SafeERC20	safeDecreaseAllowance	internal		
SafeERC20	callOptionalReturn	private		
Roles	add	internal		
Roles	remove	internal		
Roles	has	internal		view
IERC20Vestable	getIntrinsicVestingSchedule	external		view
IERC20Vestable	grantVestingTokens	external		
IERC20Vestable	currentTime	external		view
IERC20Vestable	vestingForAccountAsOf	external		view
IERC20Vestable	vestingAsOf	external		view

IERC20Vestable	revokeGrant	external		
GrantorRole	"constructor"	internal		
GrantorRole	isGrantor	public		view
GrantorRole	addGrantor	public	onlyGrantor	
GrantorRole	removeGrantor	public	onlyGrantor	
GrantorRole	_addGrantor	private		
GrantorRole	_removeGrantor	private		
GrantorRole	applyGrantor	public	onlyGrantor	
VerifiedAccount	"constructor"	internal		
VerifiedAccount	registerAccount	public		
VerifiedAccount	isRegistered	public		view
VerifiedAccount	_accountExists	internal		view
IERC20	totalSupply	external		view
IERC20	balanceOf	external		view
IERC20	transfer	external		
IERC20	allowance	external		view
IERC20	approve	external		
IERC20	transferFrom	external		
PauserRole	"constructor"	internal		
PauserRole	isPauser	public		view
PauserRole	addPauser	public	onlyPauser	
PauserRole	renouncePauser	public		
PauserRole	_addPauser	internal		
PauserRole	_removePauser	internal		

Pausable	"constructor"	internal		
Pausable	paused	public		view
Pausable	pause	public	onlyPauser, whenNotPaused	
Pausable	unpause	public	onlyPauser, whenPaused	
Ownable	"constructor"	internal		
Ownable	owner	public		view
Ownable	isOwner	public		view
Ownable	renounceOwnership	public	onlyOwner	
Ownable	transferOwnership	public	onlyOwner	
Ownable	_transferOwnership	internal		
ERC20	"constructor"	public		
ERC20	initalCapacity	public		view
ERC20	addBalance	internal		
ERC20	subtractBalance	internal		
ERC20	unitsOneEthCanBuy	public		view
ERC20	totalEthInWei	public		view
ERC20	updateTotalEthInWei	internal		
ERC20	ownerWallet	public		view
ERC20	allowed	public		view
ERC20	setAllowedAmount	internal		
ERC20	totalSupply	public		view
ERC20	balanceOf	public		view
ERC20	approve	public		
ERC20	transferFrom	public		

ERC20	transfer	public		
ERC20	_transfer	private		
ERC20	increaseAllowance	public		
ERC20	decreaseAllowance	public		
ERC20	_approve	internal		
ERC20	_burnFrom	internal		
ERC20	_burn	internal		
TokenVesting	_setVestingSchedule	internal		
TokenVesting	_hasVestingSchedule	internal		view
TokenVesting	getIntrinsicVestingSchedule	public	onlyGrantorOrSelf	view
TokenVesting	_grantVestingTokens	internal		
TokenVesting	grantVestingTokens	public	onlyGrantor	
TokenVesting	safeGrantVestingTokens	public	onlyGrantor, onlyExistingAccount	
TokenVesting	currentTime	public		view
TokenVesting	_effectiveHours	internal		view
TokenVesting	_getNotVestedAmount	public		view
TokenVesting	_getAvailableAmount	internal		view
TokenVesting	vestingForAccountAsOf	public	onlyGrantorOrSelf	view
TokenVesting	vestingAsOf	public		view
TokenVesting	_fundsAreAvailableOn	internal		view
TokenVesting	revokeGrant	public	onlyGrantor	
NexxoTokens	"constructor"	public	ERC20	
NexxoTokens	"fallback"	external	whenNotPaused	payable
NexxoTokens	totalSupply	public		view

NexxoTokens	balanceOf	public		view
NexxoTokens	allowance	public		view
NexxoTokens	transferFrom	public	whenNotPaused, onlyIfFundsAvailableNow	
NexxoTokens	transfer	public	whenNotPaused	
NexxoTokens	approve	public	onlyOwner	
NexxoTokens	burn	public	onlyOwner	
NexxoTokens	burnFrom	public	onlyOwner	
NexxoTokens	startVestingTokens	public	whenNotPaused, onlyOwner	
NexxoTokens	fetchNotVestedAmount	public	onlyOwner	view
NexxoTokens	fetchAvailableAmount	public	onlyOwner	view
NexxoTokens	getIntrinsicVestingSchedule	public	onlyOwner	view
NexxoTokens	isBlocked	internal		view
NexxoTokens	getBlockedAddressCount	public	onlyOwner	view
NexxoTokens	getBlockedAddressList	public	onlyOwner	view
NexxoTokens	blockWalletAddress	public	onlyOwner	
NexxoTokens	unblockWalletAddress	public	onlyOwner	

5.3 Modifiers

contract	modifier
GrantorRole	onlyGrantor
GrantorRole	onlyGrantorOrSelf
VerifiedAccount	onlyExistingAccount
PauserRole	onlyPauser
Pausable	whenNotPaused
Pausable	whenPaused
Ownable	onlyOwner
TokenVesting	onlyIfFundsAvailableNow

5.4 States

contract	state	type	visibility	isConst
GrantorRole	_grantors	Roles.Role	private	false
VerifiedAccount	_isRegistered	mapping	private	false
PauserRole	_pausers	Roles.Role	private	false
Pausable	_paused	bool	private	false
Ownable	_owner	address	private	false
ERC20	INITIAL_SUPPLY	uint	private	false
ERC20	_totalSupply	uint	private	false
ERC20	_balances	mapping	default	false
ERC20	_unitsOneEthCanBuy	uint256	private	false
ERC20	_totalEthInWei	uint256	private	false

ERC20	_ownerWallet	address	default	false
ERC20	_allowed	mapping	default	false
TokenVesting	THOUSAND_YEARS_HOURS	uint32	private	true
TokenVesting	TEN_YEARS_HOURS	uint32	private	true
TokenVesting	SECONDS_PER_HOUR	uint32	private	true
TokenVesting	JAN_1_2000_SECONDS	uint32	private	true
TokenVesting	JAN_1_2000_HOURS	uint32	private	true
TokenVesting	JAN_1_3000_HOURS	uint32	private	true
TokenVesting	_vestingSchedules	mapping	private	false
TokenVesting	_tokenGrants	mapping	private	false
NexxoTokens	name	string	public	false
NexxoTokens	symbol	string	public	false
NexxoTokens	decimals	uint8	public	false
NexxoTokens	INITIAL_SUPPLY	uint	private	false
NexxoTokens	UNIT_PER_ETH_BUY	uint256	private	false
NexxoTokens	blockedAddressStructs	mapping	default	false
NexxoTokens	blockedAddressList	array	default	false

6. Test Suite Results

The NEXXO Token is part of the Nexxo Smart Contract and this one was audited. All the functions and state variables are well commented using the natspec documentation for the functions which is good to understand quickly how everything is supposed to work.

6.1 Mythril Classic & MYTHX Security Audit

Mythril Classic is an open-source security analysis tool for Ethereum smart contracts. It uses concolic analysis, taint analysis and control flow checking to detect a variety of security vulnerabilities.

Detected Vulnerabilities
Informational: 0
Low: 2
Medium: 0
High: 0
Critical: 0

Issues

6.1.1 A floating pragma is set.

Severity: LOW
Code: SWC-103
File(s) affected: NexxoToken.sol

Attack / Description	Code Snippet	Result/Recommendation
The current pragma Solidity directive is ""**>=0.5.3<=0.5.8**"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary	Line: 1 `pragma solidity >=0.5.3 <=0.5.8;`	It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. Pragma solidity 0.5.3

between builds. This is especially important if you rely on bytecode-level verification of the code.		

6.1.2 Implicit loop over unbounded data structure.

Severity: LOW

Code: SWC-128

File(s) affected: NexxoToken.sol

Attack / Description	Code Snippet	Result/Recommendation
Gas consumption in function "getBlockedAddressList" in contract "NexxoTokens" depends on the size of data structures that may grow unboundedly. The highlighted statement involves copying the array "blockedAddressList" from "storage" to "memory". When copying arrays from "storage" to "memory" the Solidity compiler emits an implicit loop.If the array grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an	Line: 1438 – 1440 `function getBlockedAddressList() public onlyOwner view returns(address [] memory) { return blockedAddressList; }`	Only the Owner can use that function. The NEXXO Smart Contract is secure against that attack

attacker might attempt to cause this condition on purpose.		

Result: The analysis was completed successfully. No major issues were detected.

7. Specific Attacks

Detected Vulnerabilities
Informational: 0
Low: 0
Medium: 3
High: 1
Critical: 0

Issues

Attack / Description	Code Snippet	Severity	Result/Recommendation

Potential Violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove,TokenVesting.revoke, SafeERC20.safeIncreaseAllowance, SafeERC20.safeDecreaseAllowance(contract IERC20,address,uint256):	Line: 132 – 135
	
	`function safeIncreaseAllowance(IERC20 token, address spender, uint256 value) internal {`
	`uint256 newAllowance = token.allowance(address(this), spender).add(value);`
	`callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, newAllowance));`
	`}`
		Severity: 2	Could potentially lead to re-entrancy vulnerability.
			
			The NEXXO Smart Contract is secure against that attack

An Attack Vector on Approve/TransferFrom	Line 692 – 695	**Severity: 2**	Only use the approve function of the ERC-20 standard to change allowed

Methods
 	
Source:
https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit
	`function approve(address spender, uint256 value) public returns (bool) {`
	`  _approve(msg.sender, spender, value);`
	`  return true;`
	`  }`
			amount to 0 or from 0 (wait till transaction is mined and approved).
			The NEXXO Smart Contract is secure against that attack

Checking Outdated Libraries	All libraries are based on OpenZeppelin Framework solidity 0.5.0 https://github.com/OpenZeppelin/openzeppelin-contracts/tree/release-v2.5.0	**Severity: 2**	Recommended to migrate the contract to solidity v.0.6.0 and the used libraries. Example: Line 19 - 111 SafeMath.sol migrate to v.0.6 https://github.com/OpenZeppelin/openzeppelin-contracts/commit/5dfe7215a9156465d550030eadc08770503b2b2f#diff-b7935a40e05eeb5fe9024dc210c8ad8a * Improvement: functions in SafeMath contract overloaded to accept custom error messages. * CHANGELOG updated, custom error messages added to ERC20, ERC721 and ERC777 for subtraction related exceptions. * SafeMath overloads for 'add' and 'mul' removed. * Error messages modified.
Contract code size over limit.		**Severity: 3**	The Contract as delivered reached the 24 KB code size limit. To deploy the

Contract creation initialization returns data with length of more than 24576 bytes. The deployment will likely fails.			code you need to split your contracts into various contracts by using proxies. https://blog.polymath.network/solidity-tips-and-tricks-to-save-gas-and-reduce-bytecode-size-c44580b218e6

8. SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✅
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✅
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✅
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	❌
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✅
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✅

ID	Title	Relationships	Test Result
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✅
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✅
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✅
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✅
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✅
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✅
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✅

ID	Title	Relationships	Test Result
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✅
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✅
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✅
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✅
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✅
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✅
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✅
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✅

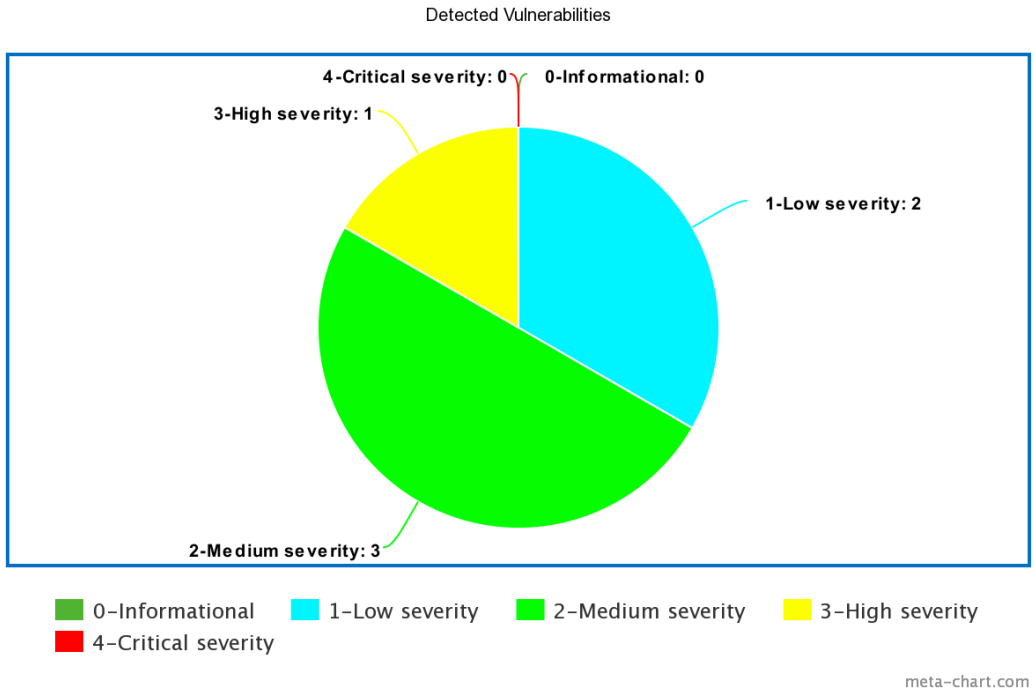
ID	Title	Relationships	Test Result
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✅
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	❌
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✅
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✅
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅

Sources:
https://smartcontractsecurity.github.io/SWC-registry
https://dasp.co
https://github.com/chainsulting/Smart-Contract-Security-Audits
https://consensys.github.io/smart-contract-best-practices/known_attacks

9. Executive Summary

A majority of the code was standard and copied from widely-used and reviewed contracts and as a result, a lot of the code was reviewed before. It correctly implemented widely-used and reviewed contracts for safe mathematical operations. The audit identified no major security vulnerabilities, at the moment of audit. We noted that a majority of the functions were self-explanatory, and standard documentation tags (such as @dev, @param, and @returns) were included.

We recommend upgrading the solidity codebase to v 0.6.0 and reduce the bytecode size.



10. Deployed Smart Contract

[To update that section, please send us the etherscan link to the deployed smart contract]