NEXXO TOKEN SMART CONTRACT AUDIT

FOR NEXXO SG PTE. LTD.

05.07.2019

Made in Germany by Chainsulting.de

Smart Contract Audit - NEXXO Token

Table of Contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, 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.

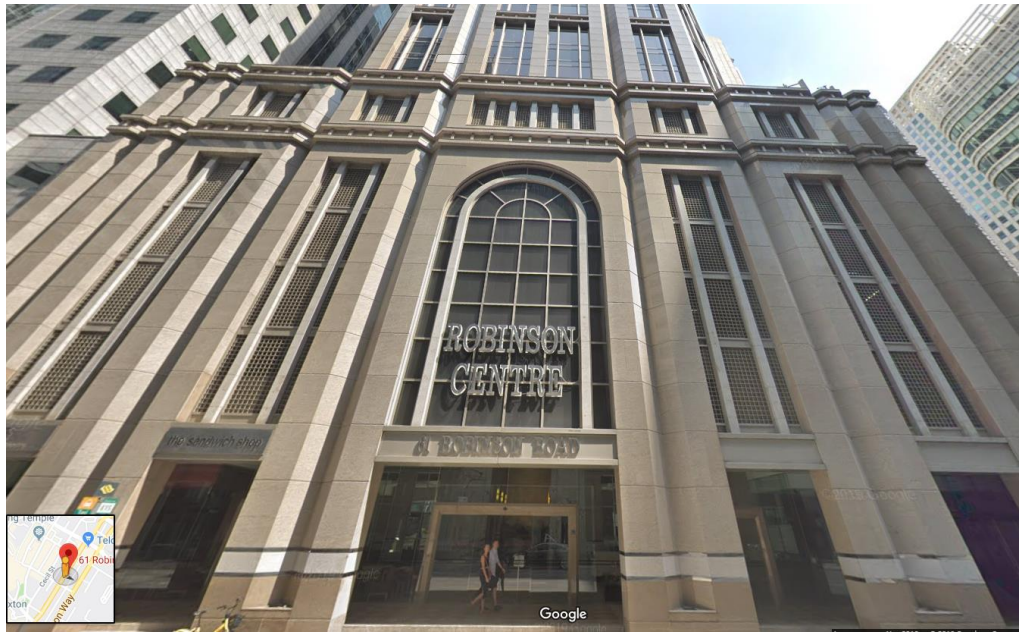
Major Versions / Date	Description	Author
0.1 (27.06.2019)	Layout	Y. Heinze
0.5 (28.06.2019)	Automated Security Testing Manual Security Testing	Y. Heinze
1.0 (01.07.2019)	Summary and Recommendation	Y. Heinze
1.1 (05.07.2019)	Deploy to Main Network Ethereum	Y. Heinze / NEXXO
1.2 (05.07.2019)	Last Security Check and adding of recommendations	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



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.

3. Vulnerability Level

0-Informational severity – A vulnerability that have informational character but is not effecting any of the code.

1-Low severity - A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

2-Medium severity – A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

3-High severity – A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.

4-Critical severity – A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.

4. Overview of the audit

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.

4.1 Used Code from other Frameworks/Smart Contracts

1. SafeMath.sol
https://github.com/OpenZeppelin/openzeppelin-solidity/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
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/token/ERC20/ERC20.sol

4. IERC20.sol
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/token/ERC20/IERC20.sol

5. Ownable.sol
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/ownership/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
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/token/ERC20/SafeERC20.sol

10. TokenVesting.sol
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/drafts/TokenVesting.sol

4.2 Tested Contract Files

File	Checksum (SHA256)
Nexxo_contract_solidity_05__0_5_3.sol	be6b71d1b84cecb0e14d41133349b955b18e0fae6c19ff9ba33ce061da3f9861

https://github.com/chainsulting/Smart-Contract-Security-Audits/blob/master/Nexxo/Nexxo_contract_solidity_05__0_5_3.sol

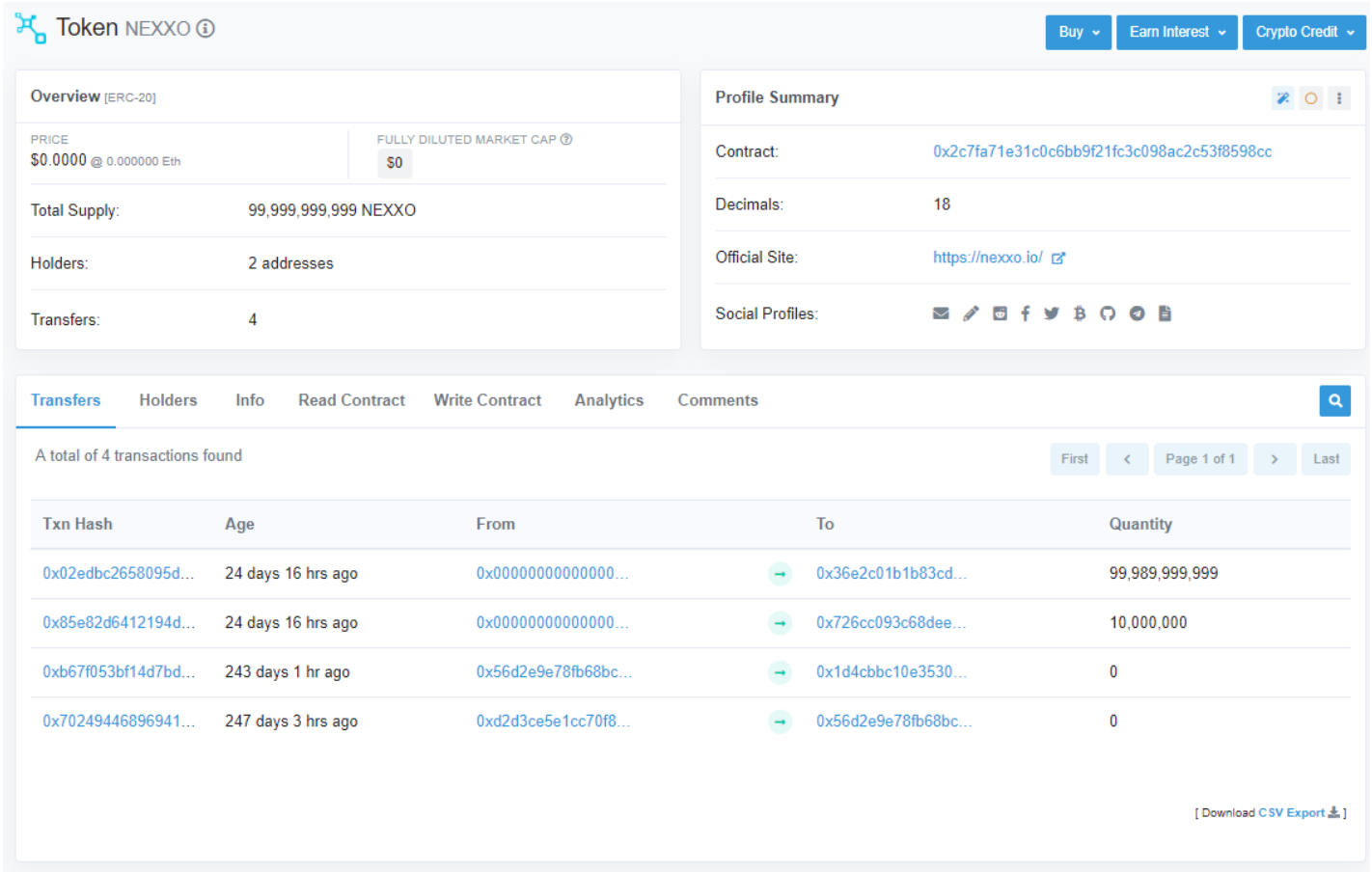
4.3 Contract Specifications

Language	Solidity
Token Standard	ERC20
Most Used Framework	OpenZeppelin
Compiler Version	0.5.3
Burn Function	Yes
Mint Function	No
Ticker Symbol	NEXXO
Total Supply	100 000 000 000
Timestamps used	Yes (blocktime)

4.4 Special Security Note

The following Smart Contract is outdated and not anymore used by Nexxo Network Company. DON’T USE IT !

https://etherscan.io/token/0x2c7fa71e31c0c6bb9f21fc3c098ac2c53f8598cc



5. Summary of Smart Contract

Functions

contract	func	visibility	modifiers	stateMutability
SafeMath	add	internal		pure
SafeMath	sub	internal		pure
SafeMath	mul	internal		pure
SafeMath	div	internal		pure
SafeMath	mod	internal		pure
Roles	add	internal		
Roles	remove	internal		
Roles	has	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	
ERC20	totalSupply	public		view
ERC20	balanceOf	public		view
ERC20	transfer	public		
ERC20	allowance	public		view
ERC20	approve	public		
ERC20	transferFrom	public		
ERC20	increaseAllowance	public		
ERC20	decreaseAllowance	public		
ERC20	_transfer	internal		
ERC20	_burn	internal		
ERC20	_approve	internal		
ERC20	_burnFrom	internal		
Address	isContract	internal		view
Address	toPayable	internal		pure
SafeERC20	safeTransfer	internal		
SafeERC20	safeTransferFrom	internal		
SafeERC20	safeApprove	internal		
SafeERC20	safeIncreaseAllowance	internal		
SafeERC20	safeDecreaseAllowance	internal		
SafeERC20	callOptionalReturn	private		
ERC20Burnable	burn	public		
ERC20Burnable	burnFrom	public		
Ownable	"constructor"	internal		
Ownable	owner	public		view
Ownable	isOwner	public		view
Ownable	renounceOwnership	public	onlyOwner	
Ownable	transferOwnership	public	onlyOwner	

Ownable	_transferOwnership	internal		
ApproveAndCallFallBack	receiveApproval	public		
TokenVesting	"constructor"	public		
TokenVesting	beneficiary	public		view
TokenVesting	cliff	public		view
TokenVesting	start	public		view
TokenVesting	duration	public		view
TokenVesting	revocable	public		view
TokenVesting	released	public		view
TokenVesting	revoked	public		view
TokenVesting	release	public		
TokenVesting	revoke	public	onlyOwner	
TokenVesting	_releasableAmount	private		view
TokenVesting	_vestedAmount	private		view
NexxoToken	"constructor"	public		
NexxoToken	"fallback"	external		payable
NexxoToken	totalSupply	public		view
NexxoToken	balanceOf	public		view
NexxoToken	transfer	public		
NexxoToken	approve	public		
NexxoToken	transferFrom	public		
NexxoToken	allowance	public		view
NexxoToken	approveAndCall	public		
NexxoToken	burn	public	onlyOwner	

modifiers

contract	modifier
PauserRole	onlyPauser
Pausable	whenNotPaused
Pausable	whenPaused
Ownable	onlyOwner

states

contract	state	type	visibility	isConst
PauserRole	_pausers	Roles.Role	private	false
Pausable	_paused	bool	private	false
ERC20	_balances	mapping	private	false
ERC20	_allowances	mapping	private	false
ERC20	_totalSupply	uint256	private	false
Ownable	_owner	address	private	false
TokenVesting	_beneficiary	address	private	false
TokenVesting	_cliff	uint256	private	false
TokenVesting	_start	uint256	private	false
TokenVesting	_duration	uint256	private	false
TokenVesting	_revocable	bool	private	false
TokenVesting	_released	mapping	private	false
TokenVesting	_revoked	mapping	private	false
NexxoToken	symbol	string	public	false
NexxoToken	name	string	public	false

NexxoToken	decimals	uint8	public	false
NexxoToken	_totalSupply	uint	public	false
NexxoToken	unitsOneEthCanBuy	uint256	public	false
NexxoToken	totalEthInWei	uint256	public	false
NexxoToken	fundsWallet	address	default	false
NexxoToken	balances	mapping	default	false
NexxoToken	allowed	mapping	default	false

6. Test Suite Results

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.

Attack	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: 615-618
	`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

Timestamp Dependence "block.timestamp" can be influenced by miners to a certain degree.
 	
Source: https://smartcontractsecurity.github.io/SWC-registry/docs/SWC-116
	Line: 921-927
	`if (block.timestamp < _cliff) {`
	`return 0;`
	`} else if (block.timestamp >= _start.add(_duration) || _revoked[address(token)]) {`
	`return totalBalance;`
	`} else {`
	`return totalBalance.mul(block.timestamp.sub(_start)).div(_duration);`
	`}`
		Severity: 0	Developers should write smart contracts with the notion that block timestamp and real timestamp may vary up to half a minute. Alternatively, they can use block number or external source of timestamp via oracles.

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

6.2 Oyente Security Audit
Oyente is a symbolic execution tool that works directly with Ethereum virtual machine (EVM) byte code without access to the high level representation (e.g., Solidity, Serpent).

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

7. Specific Attacks

Attack	Code Snippet	Severity	Result/Recommendation

An Attack Vector on Approve/TransferFrom Methods
 	
Source: https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit
	Line: 1019 - 1023
	`function approve(address spender, uint tokens) public returns (bool success) {`
	`allowed[msg.sender][spender] = tokens;`
	`emit Approval(msg.sender, spender, tokens);`
	`return true;`
	`}`
		Severity: 2	Only use the approve function of the ERC-20 standard to change allowed amount to 0 or from 0 (wait till transaction is mined and approved).
			The NEXXO Smart Contract is secure against that attack

Unchecked math: Solidity is prone to integer over- and underflow. Overflow leads to unexpected effects and can lead to loss of funds if exploited by a malicious account.	No critical mathematical functions are used	**Severity: 2**	Check against over- and underflow (use the SafeMath library). The NEXXO Smart Contract is secure against that attack

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	✅
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	✅

ID	Title	Relationships	Test Result
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	✅
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	✅

ID	Title	Relationships	Test Result
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	✅
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	✅

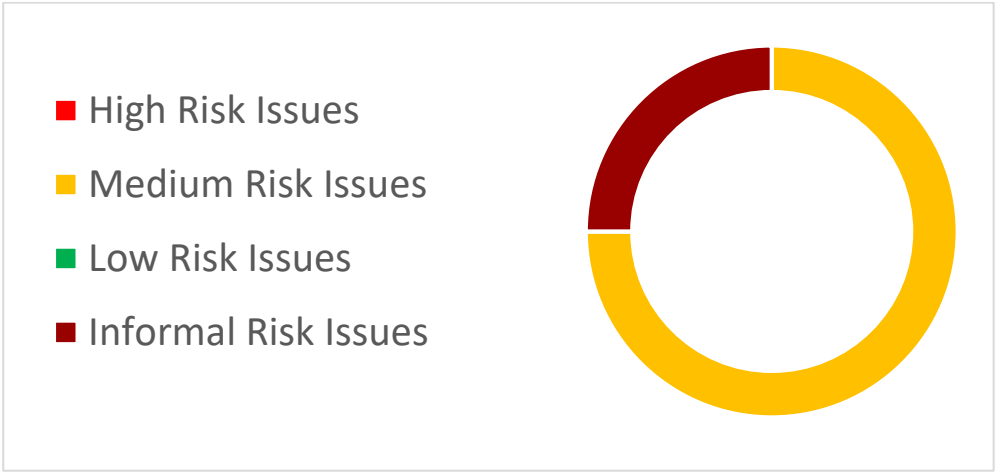
ID	Title	Relationships	Test Result
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✅
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✅
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅

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.



10. General Summary

The issues identified were minor in nature, and do not affect the security of the contract.

Additionally, the code implements and uses a SafeMath contract, which defines functions for safe math operations that will throw errors in the cases of integer overflow or underflows. The simplicity of the audited contracts contributed greatly to their security. The usage of the widely used framework OpenZeppelin, reduced the attack surface.

11. Deployed Smart Contract

0x278a83B64C3e3E1139f8E8A52D96360cA3c69A3D (NEXXO Token)

https://etherscan.io/address/0x278a83b64c3e3e1139f8e8a52d96360ca3c69a3d (Etherscan)

We recommend to update the etherscan.io information with Logo/Website/Social Media Accounts (NEXXO Token). That gives buyers more transparency.