NEXXO TOKEN SMART CONTRACT AUDIT

FOR NEXXO SG PTE. LTD.

07.08.2020

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
First Audit: https://github.com/chainsulting/Smart-Contract-Security-Audits/blob/master/Nexxo/2020/First%20Contract/02_Smart%20Contract%20Audit%20Nexxo_18_06_2020.pdf

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
1.5 (23.06.2020)	First audit review and submit changes	Y. Heinze
2.0 (29.07.2020)	Second audit review from updated contract	Y. Heinze
2.1 (07.08.2020)	Final edits and adding of the deployed contract etherscan link	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 REPRESENTATIVE: 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

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

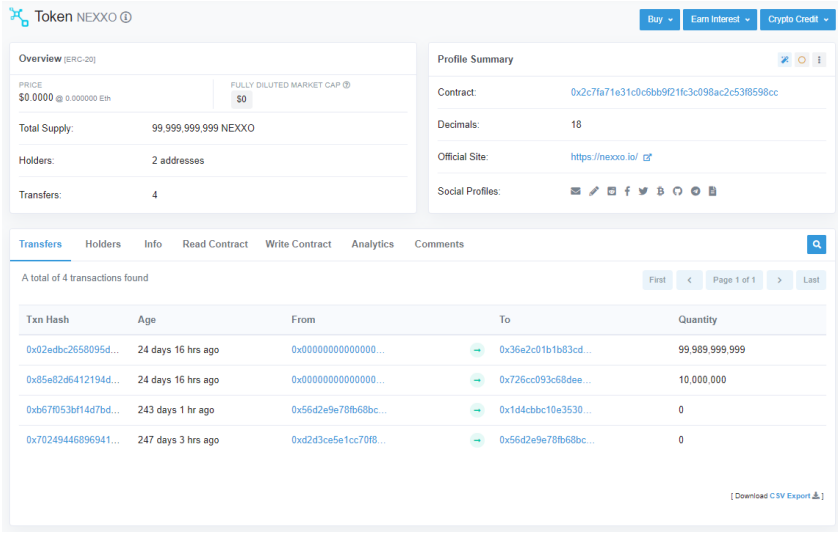
File	Fingerprint (SHA256)	Source
nexxo_contract_proxy_08_0_1.sol	AC44BBF936FBE3AF0E30A7B2CD07836A11A5770BB80C678E74F5092FE1A2E988	https://raw.githubusercontent.com/chainsulting/Smart-Contract-Security-Audits/master/Nexxo/2020/Fixed%20Contract/nexxo_contract_proxy_08_0_1.sol
nexxo_contract_solidity_08_0_1.sol	A3B521889D46A851424AC3D570104A30857647C2510B6DDA8EE4612C6E038DE5	https://raw.githubusercontent.com/chainsulting/Smart-Contract-Security-Audits/master/Nexxo/2020/Fixed%20Contract/nexxo_contract_solidity_08_0_1.sol

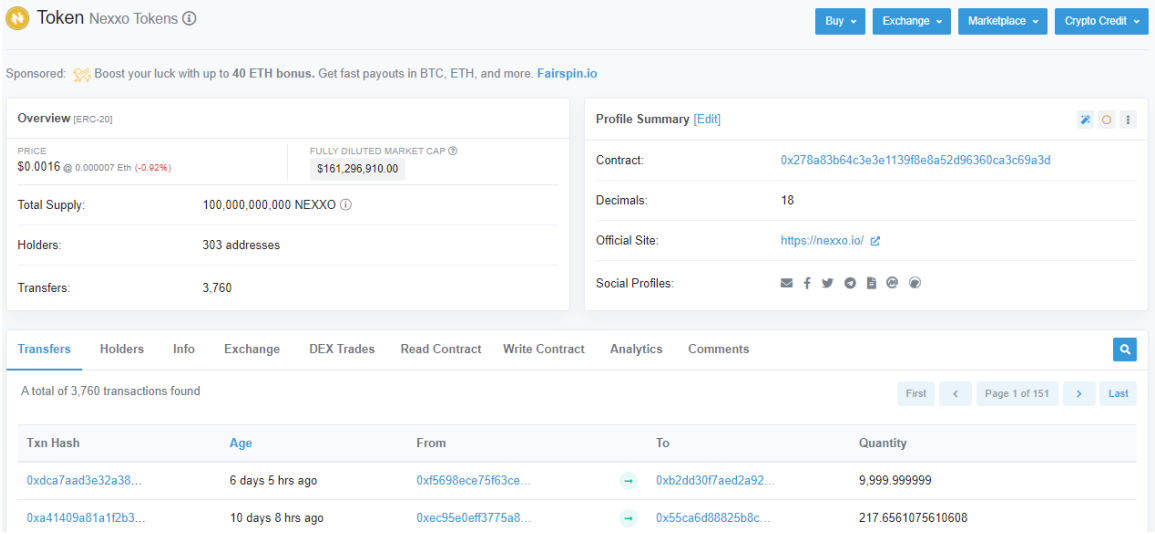
4.4 Contract Specifications

Language	Solidity
Token Standard	ERC20
Most Used Framework	OpenZeppelin
Compiler Version	0.6.11
Upgradeable	Yes
Burn Function	Yes
Proxy	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. 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.

5.1 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.

Issues (Old Contract)

Source Code: https://raw.githubusercontent.com/chainsulting/Smart-Contract-Security-Audits/master/Nexxo/2020/First%20Contract/NexxoToken.sol

5.1.1 A floating pragma is set.
Severity: LOW
Code: SWC-103
Status: Fixed
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 between builds. This is especially important if you rely on bytecode-level verification of the code.	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

5.1.2 Implicit loop over unbounded data structure.
Severity: LOW
Code: SWC-128
Status: Fixed
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 attacker might attempt to cause this condition on purpose.	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

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

6. Specific Attacks (Old Contract)

Attack / Description	Code Snippet	Severity	Result/Recommendation
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 Status: Fixed	**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. Contract creation initialization returns data with length of more than 24576 bytes. The deployment will likely fails.	Status: Fixed	**Severity: 3**	The Contract as delivered reached the 24 KB code size limit. To deploy the code you need to split your contracts into various contracts by using proxies.

7. SWC Attacks (New Contract)

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

7.1 The arithmetic operation can overflow

Severity: HIGH
Code: SWC-101
Status: Fixed (SafeMath newest version)
File(s) affected: nexxo_contract_solidity_08_0_1.sol

Attack / Description	Code Snippet	Result/Recommendation
It is possible to cause an arithmetic overflow. Prevent the overflow by constraining inputs using the require() statement or use the OpenZeppelin SafeMath library for integer arithmetic operations. Refer to the transaction trace generated for this issue to reproduce the overflow.	Line: 1013 `uint256 amount = msg.value * unitsOneEthCanBuy();`	The NEXXO Smart Contract is secure against that attack with using SafeMath library

7.2 Loop over unbounded data structure.
Severity: LOW
Code: SWC-128
File(s) affected: nexxo_contract_solidity_08_0_1.sol

Attack / Description	Code Snippet	Result/Recommendation
Gas consumption in function "toString" in contract "Strings" depends on the size of data structures or values that may grow unboundedly. If the data structure 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 attacker might attempt to cause this condition on purpose.	Line: 78 / 85 `while (temp != 0) {`	The NEXXO Smart Contract is secure against that attack

7.3 Implicit loop over unbounded data structure.
Severity: LOW
Code: SWC-128
File(s) affected: nexxo_contract_solidity_08_0_1.sol

Attack / Description	Code Snippet	Result/Recommendation
Gas consumption in function "getBlockedAddressList" in contract "NexxoTokensUpgrade1" 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 attacker might attempt to cause this condition on purpose.	Line: 1098 `return blockedAddressList;`	The NEXXO Smart Contract is secure against that attack

7.4 Call with hardcoded gas amount.
Severity: LOW
Code: SWC-134
File(s) affected: nexxo_contract_solidity_08_0_1.sol

Attack / Description	Code Snippet	Result/Recommendation

The highlighted function call forwards a fixed amount of gas. This is discouraged as the gas cost of EVM instructions may change in the future, which could break this contract's assumptions. If this was done to prevent reentrancy attacks, consider alternative methods such as the checks-effects-interactions pattern or reentrancy locks instead.	Line: 1020
	`ownerWallet().transfer(msg.value);`
	`//Transfer ether to fundsWallet`
		The NEXXO Smart Contract is secure against that attack

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

8. 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. All recommendations from the first audit are implemented by the Nexxo Team.

9. Deployed Smart Contract

Token Address:
https://etherscan.io/token/0xd98bd7bbd9ca9b4323448388aec1f7c67f733980

Contract Address:
https://etherscan.io/address/0xcabc7ee40cacf896ca7a2850187e1781b05f09c5

Proxy Contract Address:
https://etherscan.io/address/0xd98bd7bbd9ca9b4323448388aec1f7c67f733980