New Economic Coin [NEC]

SMART CONTRACT AUDIT
26th August 2023

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, 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 NEC. 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
0.1 (16.03.2022)	Layout
0.4 (16.03.2022)	Automated Security Testing
	Manual Security Testing
0.5 (17.03.2022)	Verify Claims and Test Deployment
0.6 (17.03.2022)	Testing SWC Checks
0.9 (17.03.2022)	Summary and Recommendation
1.0 (17.03.2022)	Final document

2. About the Project and Company NEC

Website: https://newainternational.org/

Facebook: https://www.facebook.com/profile.php?id=100091511306654&mibextid=ZbWKwL

Instagram: https://instagram.com/newa.international

Twitter: https://twitter.com/NEWA2023

Telegram: https://t.me/NEWA_INTERNATIONAL

YouTube: https://www.youtube.com/channel/UCf5GzM1FqLg5WyYeMkrf3FQ

Whatsapp: https://wa.me/+22994385454

Support: support@newinternational.org

Project Overview

NEWA INTERNATIONAL is a non-governmental organization (NGO) committed to the promotion of farming techniques, New Information and

Communication Technologies (NTIC) as well as to supporting national initiatives in the digital field.

We aim to actively contribute to the socio-economic development of Benin by intelligently using new technologies.

Our mission is to promote poverty reduction and improve social well-being by implementing projects and initiatives in various fields.

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.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   Review of the specifications, sources, and instructions provided to #TheBlockCodes to make sure we understand the size,scope, and functionality of the smart contract.

   Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.

   Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to #TheBlockCodes describe.

2. Testing and automated analysis that includes the following:
   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.
   Symbolic execution, which is analyzing 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.
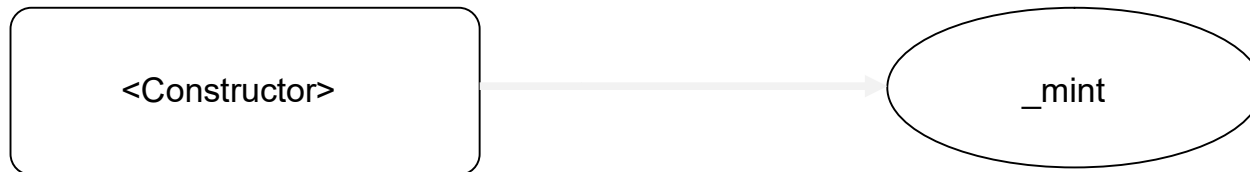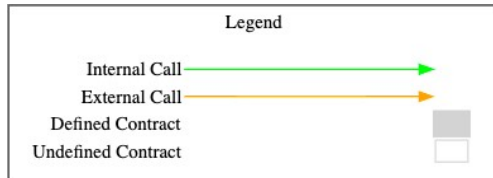
Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 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 (MD5)
contracts/NEC.sol	71zs914543j3y924hd21fd2324d34254

Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source
@openzeppelin/contracts/token/ERC20/ERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.5.0/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts@4.7.3/token/ERC20/extensions/ERC20Burnable.sol	https://github.com/openzeppelin/contracts@4.7.3/token/ERC20/extensions/ERC20Burnable.sol
@openzeppelin/contracts@4.7.3/token/ERC20/extensions/ERC20Snapshot.sol	https://github.com/openzeppelin/contracts@4.7.3/token/ERC20/extensions/ERC20Snapshot.sol
@openzeppelin/contracts@4.7.3/access/Ownable.sol	https://github.com/openzeppelin/contracts@4.7.3/access/Ownable.sol
@openzeppelin/contracts@4.7.3/token/ERC20/extensions/draft-ERC20Permit.sol	https://github.com/openzeppelin/contracts@4.7.3/token/ERC20/extensions/draft-ERC20Permit.sol
@openzeppelin/contracts@4.7.3/token/ERC20/extensions/ERC20Votes.sol	https://github.com/openzeppelin/contracts@4.7.3/token/ERC20/extensions/ERC20Votes.sol

Metrics / CallGraph

Legend

Internal Call	———————▶ (green)
External Call	———————▶ (orange)
Defined Contract	▢ (filled)
Undefined Contract	▢ (empty)

<Constructor> ——————▶ _mint

Metrics / Source Lines & Risk
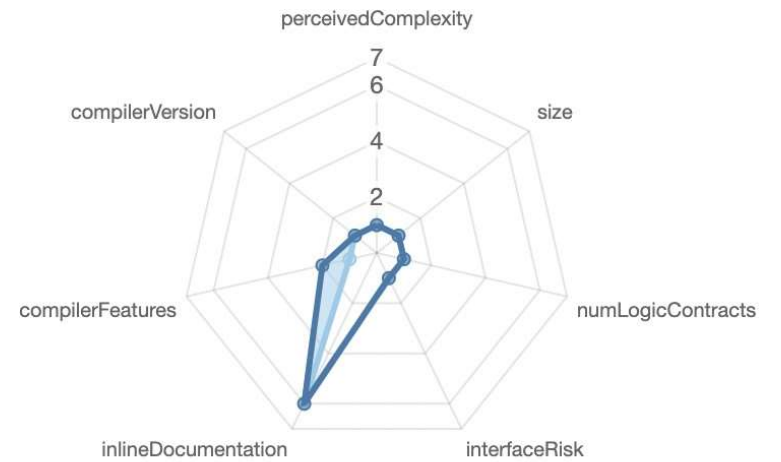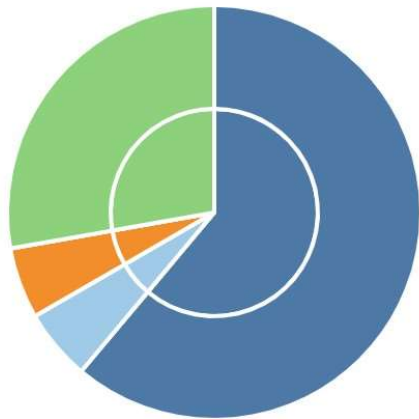
Metrics / Capabilities

Solidity Versions observed	✏️ Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`0.4.7.3`				

📤 Transfers ETH	⚡Low-Level Calls	👥 DelegateCall	🔢Uses Hash Functions	🗒️ ECRecover	🌀 New/Create/Create2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
0	0

External	Internal	Private	Pure	View
0	1	0	0	0

StateVariables

Total	🌐Public
0	0

Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	NEC.sol	1	————	52	52	6	1	5	————
📝	**Totals**	**1**	————	**52**	**52**	**6**	**1**	**5**	

Legend: [━]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

5. Scope of Work

Following contracts with the direct imports has been tested:
- o NEC.sol

The audit team put forward the following assumption regarding the security and usage of the contract:

- The contract is using the ERC-20 token standard
- Owner/Deployer cannot mint new tokens
- Owner/Deployer cannot burn or lock user funds
- Owner/Deployer cannot pause the contract
- The smart contract is coded according to the newest standards and in a secure way

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.
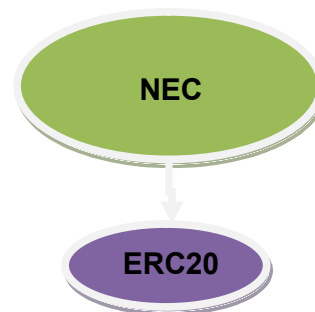
Manual and Automated Vulnerability Test

CRITICAL ISSUES
During the audit #TheBlockCode's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES
During the audit, #TheBlockCodes's found **no High issues** in the code of the smart contract.

MEDIUM ISSUES
During the audit,#TheBlockCodes's experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES
During the audit,#TheBlockCodes's experts found **no Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES
During the audit,#TheBlockCodes's experts found **no Informational issues** in the code of the smart contract.

SWC Attacks

During the audit, #TheBlockCodes's experts found **no SWC Attacks** in the code of the smart contract.

Verify Claims

The contract is using the ERC-20 token standard
 Status: tested and verified ✅

Owner/Deployer cannot mint new tokens
 Status: tested and verified ✅

Owner/Deployer cannot burn or lock user funds
 Status: tested and verified ✅

Owner/Deployer cannot pause the contract
 Status: tested and verified ✅

The smart contract is coded according to the newest standards and in a secure way.
 Status: tested and verified ✅

6. Executive Summary

Two (2) independent #TheBlockCodes experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefstook place on the August 25, 2023. The overall code quality is good and not overloaded with unnecessary functions; these are greatly benefiting the security of the contract. It correctly implemented widely used and reviewed contracts from OpenZeppelin.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work. During the audit, no issues were found after the manual and automated security testing.

7. Deployed Smart Contract

VERIFIED

https://bscscan.com/address/0xe12e375aebbc9fc1bfb17fd4e3e9af49c3e24c65#code

8. About the Auditor

#TheBlockCodes is a professional software development/Web2.0/Web3.0/MLM website/web application development firm based in India that provides comprehensive distributed ledger technology (DLT) solutions. Some of their services include blockchain development, smart contract audits and consulting.

#TheBlockCodes conducts code audits on market-leading blockchains such as Hyperledger, Tezos, Ethereum, Binance Smart Chain, and Solana to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secure the smart contracts of 1Inch, POA Network, Unicrypt, Amun, Furucombo among numerous other top DeFi projects.

#TheBlockCodes currently secures $100 billion in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the blockchain sector to deliver top-notch smart contract audit solutions tailored to the clients' evolving business needs.

The blockchain security provider brings the highest security standards to crypto and blockchain platforms, helping to foster growth and transparency within the fast-growing ecosystem.

Check our website for further information: https://theblockcodes.com

How We Work

1 --------

PREPARATION
Supply our team with audit ready code and additional materials

2 --------

COMMUNICATION
We setup a real-time communication tool of your choice or communicate via e-mails.

3 --------

AUDIT
We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.

4 --------

FIXES
Your development team applies fixes while consulting with our auditors on their safety.

5 --------

REPORT
We check the applied fixes and deliver a full report on all steps done.