NonceAudits

Coorest

CoorestOfficialToken.sol

Smart Contract Audit

Date: 17th February, 2022

www.nonceAudits.com

Introduction

This Audit Report mainly focuses on the overall security of the CoorestOfficialToken.sol contract. With this report, we have tried to ensure the reliability and correctness of their smart contract by a complete and rigorous assessment of their system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied

The Nonceblox team has performed rigorous analysis of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is well structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract to find any potential issues like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In Automated Testing, We tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was tested and this included -

- Analyzing the complexity of the code in-depth and detailed, manual review of the code, line-by-line.
- Analyzing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analyzing the security of the on-chain data.

Audit Details

Project Name: **Coorest**

Languages: Solidity (Smart contract)
Platforms and Tools: Remix IDE, Solhint, VScode, Slither, Mythril
Commit hash:32d11baaca760df5f4e44779e82ef8c99b212477

Audit Goals

The focus of the audit was to verify that the Smart Contract System is secure, resilient, and working according to the specifications. The audit activities can be grouped into the following three categories:

Security

Identifying security-related issues within each contract and the system of contract.

Sound Architecture

Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

Code Correctness and Quality

A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity

Security

Every issue in this report was assigned a severity level from the following:

High-level severity issues

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium level severity issues

Issues on this level could potentially bring problems and should eventually be fixed.

Low-level severity issues

Issues on this level are minor details and warnings that can remain unfixed but would be better fixed.

Informational-level severity issues

Issues on this level are informational details that can remain unfixed but would be better fixed.

Number of issues per severity

	INFORMATIONAL	LOW	MEDIUM	HIGH	Recommendations
OPEN	0	0	0	0	0
CLOSED	0	0	0	0	0
ACKNOWLEDGED	0	0	0	0	0

Manual Audit:

For this section, the code was tested/read line by line by our developers. We also used Remix IDE's JavaScript VM to test the contract functionality.

Low-level severity issues

- None

Medium level severity issues

- None

High-level severity issues

- None

Informational-level severity issues

- None

Automated Audit

Solhint Linting Violations

Solhint is an open-source project for linting solidity code, providing both security and style guide validations. It integrates seamlessly into most mainstream IDEs. We used Solhint as a plugin within our Remix IDE for this analysis. Several violations were detected by Solhint, it is recommended to use Solhint's npm package to lint the contract.

Mythril

Mythril is a security analysis tool for EVM bytecode. It detects security vulnerabilities in smart contracts built for Ethereum, Hedera, Quorum, Vechain, Roostock, Tron and other EVM-compatible blockchains. It uses symbolic execution, SMT solving and taint analysis to detect a variety of security vulnerabilities.

```
The analysis was completed successfully. No issues were detected.
```

Mythril didn't detect any issues.

High-level severity issues

- None

Medium-level severity issues

- None

Low-level severity issues

- None

Informational-level severity issues

- None

Slither

Slither, an open-source static analysis framework. This tool provides rich information about Ethereum smart contracts and has critical properties. While Slither is built as a security-oriented static analysis framework, it is also used to enhance the user's understanding of smart contracts, assist in code reviews, and detect missing optimizations.

High-level severity issues

- None

Medium-level severity issues

- None

Low-level severity issues

- None

Disclaimer

Nonceblox audit is not a security warranty, investment advice, or an endorsement of the CoorestOfficialToken.sol contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Summary

The use case of the smart contract is simple and the code is relatively normal. Altogether, the code is written and demonstrates effective use of abstraction, separation of concerns, and modularity.