reactive × PESSIMISTIC

# SECURITY ANALYSIS

by Pessimistic

This report is public

**January 15, 2025**

## ABSTRACT

In this report, we consider the security of smart contracts of Reactive Network Token project. Our task is to find and describe security issues in the smart contracts of the platform.

## DISCLAIMER

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

## SUMMARY

In this report, we considered the security of Reactive Network Token smart contracts. We described the audit process in the section below.

The audit showed only one issue of medium severity Excessive owner's role and one issue of low severity.

After the initial audit, the developers added the multisig wallet as the owner and addressed the issue of Excessive owner's role.

The project is already deployed.

## GENERAL RECOMMENDATIONS

We do not have any further recommendations.

# PROJECT OVERVIEW

## Project description

For the audit, we were provided with Reactive Network Token project on a public GitHub repository, commit 92cded6cabedb4f6423e170b6b24a3ba6fa91a65.

The token was already deployed at the address 0x817162975186d4d53dbf5a7377dd45376e2d2fc5. Most part of the code consists of OpenZeppelin contracts.

The scope of the audit included everything.

The documentation for the project included the description in the code, developers' comments and the following link.

All 5 tests pass successfully. The code coverage is 73.86%.

The total LOC of audited sources is 52.

# AUDIT PROCESS

We started and finished the audit on January 13, 2025.

We inspected the materials provided for the audit.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- Standard Solidity issues;
- Compliance with the ERC-20 standard;
- The difference between Ethereum and the Reactive Network.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit.

The developers addressed #M01 on January 15, 2025. We checked the provided multisig address and updated the report.

# MANUAL ANALYSIS

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

## Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

**The audit showed no critical issues.**

## Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

### M01. Excessive owner's role (addressed)

In the current implementation, the system depends heavily on the owner's role. The owner can authorize minters to mint or burn tokens in the **WrappedReact** contract unlimitedly.

Thus, certain scenarios can lead to undesirable consequences for the project and its users, such as if the owner's private keys become compromised.

We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> The developers changed the owner's address to the multisig contract at 0x05fDF59Ce5d50AB139bD0CcE571Af7c9b5F7CE0C address.

## Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

### L01. Unnecessary comparison with true

Consider replacing the `!authorizedMinters[_msgSender()] == true` check with `!authorizedMinters[_msgSender()]` to improve code readability in the `AuthorizedMinter.__checkAuthorizedMinter` function.

This analysis was performed by Pessimistic:

**Daria Korepanova**, Senior Security Engineer
**Oleg Bobrov**, Security Engineer
**Irina Vikhareva**, Project Manager
**Alexander Seleznev**, CEO

**January 15, 2025**