Aquaris Token Security Analysis

by Pessimistic

This report is public

17 September, 2021

Abstract

In this report, we consider the security of smarts contracts of Aquaris 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. One 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, security audit is not an investment advice.

Summary

In this report, we considered the security of Aquaris token smart contract. We performed our audit according to the procedure described below.

The audit discovered no issues due to code simplicity and correct use of OpenZeppelin token implementation.

General recommendations

We recommend following best practices for future development, like adding proper dependency management, tests, documentation and NatSpec comments. Also consider setting up CI to automatically run linters, security tools, and calculate code coverage.

Project overview

Project description

Initially we have been provided with Aquaris token code in form of flatten file `flattened_code.sol`, SHA1 checksum 2D9E6CF25262F11DC0045CFCF0BFE0252552F672. The audit was performed on `Aquaris.sol` file, SHA1 checksum 331FB708ADD827EB504F050EBB1BFDE91FE6F600. Two versions of the code only differ with token's name and symbol constants.

The code is not supplied with project files or tests, and has no README.md file. However, we don't treat it as an issue due to the compactness of the code.

The token is based on ERC20 implementation from OpenZeppelin library v4.2.0

The total LOC of audited sources is 8.

Token details:	
Name:	Aquaris
Symbol:	AQS
Decimals:	18
Total Supply:	500 000 000

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
  - We scan project's code base with automated tools: Slither and SmartCheck.
  - We manually verify (reject or confirm) all the issues found by tools.
- Manual audit
  - We manually analyze code base for security vulnerabilities.
  - We assess overall project structure and quality.
- Report
  - We reflect all the gathered information in 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 often lead to the loss of funds or other catastrophic failures. The contracts should not be deployed before these issues are fixed. We highly recommend fixing them.

The audit showed no critical issues

Medium severity issues

Medium issues can influence project operation in 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.

The audit showed no medium severity issues

Low severity issues

Low severity issues don't directly affect project's operations. However, they might lead to various problems in the future versions of the code. We recommend taking them into account.

The audit showed no low severity issues

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Vladimir Tarasov, Security Engineer
Alexander Seleznev, Founder

17 September, 2021