A2DAO Security Analysis

by Pessimistic

This report is private.

Published: March 11, 2021

Abstract

In this report, we consider the security of smart contracts of A2DAO 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 A2DAO smart contracts. We performed our audit according to the procedure described below.

The audit only showed a few issues of low severity. However, the project has no documentation and tests coverage is insufficient, which are considered medium severity issues.

The token contract is pausable.

General recommendations

We recommend adding documentation to the project. In case if developers decide to redeploy the contract, we highly recommend fixing the mentioned issues, monitoring and improving tests coverage.

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.

Project overview

Project description

For the audit, we were provided with A2DAO project on a private GitHub repository, commit 23589a01e71e8d9a8d628efdf495d85cd4c1f708.

The project has no documentation.

The total LOC of audited sources is 184.

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 smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in current implementation. We highly recommend addressing them.

No documentation

The project has no documentation. The documentation helps auditors to verify the correctness of the code. It also helps other developers with token integration.

We strongly recommend adding public documentation to the project.

Low tests coverage

The provided code has tests. However, the coverage is below 40% and is not measured regularly.

Testing is crucial for code security and audit does not replace tests in any way.

We highly recommend checking and improving test coverage regularly as the project grows.

Low severity issues

Low severity issues can influence project operation in future versions of code. We recommend taking them into account.

Code quality

- Consider using low-level `.call()` for sending ether in `withdrawCollectedEth()` and `sale()` functions of **A2Sale** contract, lines 110 and 66, respectively. Since these functions are declared as `nonReentrant`, the usage of `.call()` is safe and helps to avoid `Out of Gas` issues.
- `withdrawToken()` function of **A2Sale** contract has `nonReentrant` modifier. In this case, the modifier is redundant. Consider removing it to optimize gas consumption.
- Consider specifying the visibility of constants at lines 18–21 explicitly in **A2Token** contract to improve code readability.

Dependency management

We recommend removing **package-lock.json** file since `yarn` is used as a package manager for the project.

Notes

Pausable contract

A2Token contract can be paused at any moment by `PAUSE_MANAGER_ROLE`.

We recommend using proper key management system for this role, e.g. multisig.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Vladimir Tarasov, Security Engineer
Boris Nikashin, Analyst
Alexander Seleznev, Founder

March 11, 2021