DEXDAO

Dexdao Security Analysis

by Pessimistic

This report is public

July 27, 2023

Abstract

In this report, we consider the security of smart contracts of Dexdao 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 Dexdao smart contracts. We described the audit process in the section below.

The initial audit showed one critical issue: Liquidity pool price manipulation. The audit also revealed one issue of medium severity: (DOS because of create2). Moreover, several low-severity issues were found.

After the initial audit, the codebase was updated. All issues were fixed. No new issues were found.

After the discussion with developers, one issue of medium severity (DOS because of create2, M01) was removed from the report as it was identified as a false positive.

The overall code quality is good. The code is covered with NatSpec comments, and the developers follow the best practices.

General recommendations

We have no further recommendations.

Project overview

Project description

For the audit, we were provided with Dexdao project on a private GitHub repository, commit 8941bc2b4c0cbc41f1caaff296c0f3cbe54045af.

The scope of the audit included `contracts/` folder.

The documentation for the project included `README.md`.

All 15 tests passed successfully. The code coverage was 100%.

The total LOC of audited sources was 902.

Codebase update

After the initial audit, the codebase was updated. We were provided with the commit a6979eb0c9a2a2ff38d579ab0a2651bffe3d79d9. This update contained fixes and comments for all issues.

All 15 tests passed successfully. The code coverage was 100%.

Audit process

We started the audit on June 30 and finished on July 7, 2023.

We inspected the materials provided for the audit. Then, we contacted the developers for an introduction to the project.

During the work, we stayed in touch with the developers and discussed confusing or suspicious parts of the code.

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:

- Whether it is possible to manipulate the token price;
- Whether the integration with Uniswap V3 is correct;
- Whether the formulas correspond to the project description;
- Whether it is possible to steal the funds.

We scanned the project with the static analyzer Slither and our plugin Slitherin with an extended set of rules and manually verified the output.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

After the initial audit, we discussed the results with the developers. On July 13, the developers provided us with an updated version of the code.

We reviewed the updated codebase and confirmed the fixes.

Moreover, we re-run the tools, tests and calculated the code coverage.

Finally, we 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.

C01. Liquidity pool price manipulation (fixed)

If a malicious user holds tokens before `provideLiquidity` was invoked, they can exploit the situation by self-providing liquidity, inflating token prices, and quickly selling their tokens at a profit after the invocation. The user cannot get the tokens from the `Launchspace`, until the liquidity is provided through `provideLiquidity`. Nevertheless, there is one scenario in which tokens can be obtained in advance. If the token has an underlying token, it becomes possible to wrap the underlying tokens to the actual tokens after a 30-day period. If the sale was initiated at least 30 days prior to the `saleStartTimestamp`, the user will have sufficient time to get the tokens.

The issue has been fixed and is not present in the latest version of the code.

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.

The audit showed no issues of medium severity.

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. Division by zero (fixed)

If `minPrice` is set as zero during setup sale, `currentPrice`, and `newPrice` are zero too. As a result, their average value (`buyPrice`) is also zero. It leads to division by zero at line 203 in the `buyParams` function of the **Launchspace** contract. We recommend checking `minPrice` for the zero value.

The issue has been fixed and is not present in the latest version of the code.

L02. Unchecked zero address (fixed)

When a manager creates a token without an underlying token, they spend extra gas at lines 88–93 in the `createToken` function of the **TokenFactory** contract because of the **TokenWrapper** deploy. Consider adding an additional check whether the underlying token address is zero and deploy according to this check.

The issue has been fixed and is not present in the latest version of the code.

L03. Magic numbers (fixed)

We recommend adding comments to the magic numbers `-887_200` and `887_200` in the `provideLiquidity` of the **Launchspace**, explaining that they are the least and most possible ticks (preferably with the link to the Uniswap documentation). This will increase the code readability.

The issue has been fixed and is not present in the latest version of the code.

Notes

N01. Overpowered roles (commented)

The project highly relies on the roles which allow upgrading the logic of the contracts.

> *Comment from the developers: In the future, these roles will be managed by the DAO.*

This analysis was performed by Pessimistic:

Yhtyyar Sahatov, Security Engineer
Daria Korepanova, Senior Security Engineer
Irina Vikhareva, Project Manager

July 27, 2023