azuro × PESSIMISTIC

SECURITY ANALYSIS

by Pessimistic

This report is public

November 5, 2024

ABSTRACT

In this report, we consider the security of smart contract of Azuro project. Our task is to find and describe security issues in the smart contract 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 Azuro CashOut smart contracts. We described the audit process in the section below.

The audit showed one issue of medium severity: Documentation. Also, one low severity issue was found, and two notes were added.

GENERAL RECOMMENDATIONS

We recommend fixing the mentioned issues and improving NatSpec coverage and documentation.

PROJECT OVERVIEW

Project description

For the audit, we were provided with Azuro CashOut project on a private GitHub repository, commit a7af77824c9e0887511c8b51b37021e2bd170d55.

The scope of the audit included **contracts/extensions/CashOut.sol** smart contract.

There was no documentation for the project.

470 tests out of all 471 passed successfully; one test was pending. For the given scope, all 35 tests from **test/cash-out-test.js** passed. The code coverage could not be calculated due to the size of the repository.

The total LOC of audited sources is 206.

AUDIT PROCESS

We started the audit on October 31, 2024 and finished on November 1, 2024.

We inspected the materials provided for the audit. We performed preliminary research and specified those parts of the code and logic that require additional attention during an audit:

- Whether it is not possible to redeem bets for an unauthorized user;
- Whether signatures are checked correctly;
- Whether all the required data is signed;
- Whether checks for bets are performed correctly.

We manually analyzed all the contracts within the scope of the audit and checked their logic. We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts. We also sent the results to the developers in the text file.

We ran tests and tried to calculate the code coverage.

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

MANUAL ANALYSIS

The contracts were completely manually analyzed, and 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. Documentation

There was no documentation for the audited smart contract besides some comments from the developers.

The documentation is a critical part that helps to improve security and reduce risks. It should explicitly explain the purpose and behavior of the contracts, their interactions, and key design choices. It is also essential for any further integrations.

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. Implicit visibility

`betTokens` and `payoutToken` have implicit `inner` visibility. To improve code readability, consider writing visibility explicitly at lines 90 and 93 of the **CashOut** contract.

Notes

N01. No method for bets redeeming

There is no way to transfer or redeem bought bets. Consider adding a function to allow this functionality in the **CashOut** contract.

N02. Project roles

The system relies heavily on the `owner` role. Owner can:

- `pause`/`unpause` contract;
- Upgrade contract;
- Change accepting betting tokens;
- Change oracle;
- Withdraw **ERC20** tokens;
- Potentially redeem bought bets.

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

This analysis was performed by Pessimistic:

Daria Korepanova, Senior Security Engineer
Oleg Bobrov, Security Engineer
Irina Vikhareva, Project Manager

November 5, 2024