Azuro Security Analysis

by Pessimistic

This report is public

April 29, 2022

Abstract

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

Disclaimer

Summary

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

The initial audit showed several issues of medium severity, including Overpowered roles and Affiliate address manipulation, and several low-severity issues. Some elements of contract system architecture are mediocre.

After the initial audit, the codebase was updated. In this update, the developers fixed some of the previously discovered issues. However, several new issues were found.

After the first recheck, the codebase was updated again. This update had fixes for issues from the previous recheck, including Affiliate address manipulation, Insufficient documentation, and the new functionality. However, Incorrect integration with OpenZeppelin medium and several low severity issues were discovered.

General recommendations

We recommend fixing the mentioned issues.

Project overview

Project description

For the audit, we were provided with Azuro project on a private GitHub repository, commit 821831d8e181863019aaab7bae6e25f8c5053b80.

The documentation for the project is available by the link. The codebase has some NatSpec comments. However, we consider available documentation insufficient.

All 52 tests pass. The overall code coverage is 88.28%.

The total LOC of audited sources is 1096.

Codebase update #1

After the initial audit, we were provided with commit 0c9ee668b87901bd127ed1231b7a1ea04658ffb3.

In this update, the developers fixed most of the issues of low severity, added new functionality and new tests. However, in the updated codebase, we discovered new issues. The code coverage increased to 98,47%. All 66 tests pass.

Codebase update #2

After the first recheck the codebase was updated again. This time we were provided with commit 87a70a6d831806a9fcdcc5052bd0d1f6a7113d4d on a private GitHub repository.

In this update, the developers fixed two medium and several low severity issues, added NatSpecs, added `LiquidityTree` contract and changed the number of tests. All 54 tests pass. However, in the updated codebase, we discovered new issues.

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 the project's codebase with the automated tool Slither.
  - We manually verify (reject or confirm) all the issues found by the tool.
- Manual audit
  - We manually analyze the codebase for security vulnerabilities.
  - We assess the 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 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 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.

Affiliate address manipulation (fixed)

In **LP** contract, `bet` function does not verify affiliate address passed as a parameter. One can provide any address, e.g., their own, which gives an unfair advantage to the user.
Update 2: The issue has been fixed since this part of the code has been removed from the codebase.

Insufficient documentation (fixed)

The project has documentation. However, this documentation does not cover some important details of the project operation. Also, a significant part of the codebase is not covered with NatSpec comments.

The documentation is a critical part of the project 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.
Update 2: The issue has been fixed and is not present in the latest version of the code.

Overpowered roles

The project has special roles with excessive powers. As a result, the system depends heavily on these roles.

In **Core** contract, the `maintainer` role can:

- Cancel conditions.
- Change the expiration date of any condition.
- Update reinforcements.
- Update margins.

In **Core** contract, the `owner` can:

- Change **LP** address.
- Add/remove oracles and maintainers.

In **LP** contract, the `owner` can:

- Change **Core** address.
- Change **AzuroBet** address.
- Change fees, including the one that is transferred to the owner.

In **AzuroBet** contract, the `owner` can:

- Change **LP** address.

We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.
Update 2: The issue has been updated since some part of the code has been removed from the codebase.

Incorrect integration with OpenZeppelin (new)

Function `_beforeTokenTransfer` of **AzuroBet** contract at line 96 should be `virtual` and call `super._beforeTokenTransfer(from, to, amount)` as described in OpenZeppelin documentation.

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.

Code quality

- In **AzuroBet** contract, `tokenURI` function returns a fixed string with a confusing value at line 73. Consider adding comments and explaining this value in the documentation.
  Update 1: The issue has been fixed and is not present in the latest version of the code.

- Consider declaring `liquidityRequest` and `changePeriod` functions of **LP** contract as `external` instead of `public` to improve code readability and optimize gas consumption.
  Update 1: The issue has been fixed and is not present in the latest version of the code.

- Consider emitting events when changing important parameters of the system in:
  - **LP** contract at lines 117, 121, 125, and 397. *Update 1: The issue has been fixed and is not present in the latest version of the code.*
  - **Core** contract at lines 346, 351, 355, 359. *Update 1: The issue has been fixed and is not present in the latest version of the code.*
  - **Core** contract `changeMaxBanksRatio` function.

- In **LP** contract, consider declaring `newsavedPeriods` variable from line 392 a bit earlier and reusing its value at line 390 within the `newinitStartDate` declaration.
  Update 1: The issue has been fixed and is not present in the latest version of the code.

- In **LP** contract, the check at line 205 is redundant:

  ```
  amountLP <= requests[_getCurPeriod()].totalValue
  ```

  This check always passes if the first condition is true.
  Update 1: The issue has been fixed and is not present in the latest version of the code.

- In **Core** contract, consider declaring numeric value `10000` as a constant at lines 189 and 191.
  Update 1: The issue has been fixed and is not present in the latest version of the code.

- In `_viewPayout` function of **Core** contract, two code at lines 510-524 includes the same logic twice. Consider combining these `if` conditions into a single block to avoid code duplication.
  Update 1: The issue has been fixed and is not present in the latest version of the code.

- The following functions violate the [CEI (checks-effects-interactions) pattern]:
  - `claimDAOReward` function of **LP** contract.
  - `addLiquidity` function of **LP** contract.
  - `bet` function of **LP** contract.
  - `createCondition` function of **Core** contract.

  We highly recommend following CEI pattern to increase the predictability of the code execution and protect from some types of re-entrancy attacks.
  Update 1: The issues have been fixed and are not present in the latest version of the code.

- `available` variable is not used in `workingReserve` function of **LP** contract. Consider leaving only the type `uint256` of returned value.
  Update 2: The issue has been fixed since this part of the code has been removed from the codebase.

- (new) Consider declaring the address parameter of `withdrawn` event in **LiquidityTree** contract as indexed.

- (new) The internal functions `nodeWithdraw`, `add` and `remove` of **LiquidityTree** contract are not used.

Contract system architecture

Some design decisions are suboptimal:

- **Core** and **LP** contracts are tightly coupled, E.g., the **Core** includes affiliates logic, which has little to do with the betting math.
  Update 2: The issue has been fixed since this part of the code has been removed from the codebase.

- **LP** contract holds assets and includes the logic of both liquidity providers and gamblers. Consider splitting it into two parts.

On the other side, using NFT tokens to represent individual bets is a great choice.

Gas consumption

- In **Core** contract, `createCondition` function initializes fields of `newCondition` struct consequently at lines 137–154. Consider declaring the struct within a single statement to optimize gas consumption and prevent some types of errors.
  Update 2: The lines of this part of the code have been changed to 127-142. It has also become relevant to `Bet` structure in `putBet` function at lines 194-199.

- Optimizer `runs` parameter is set to 1, which is suboptimal for frequently called functions like `bet`. Consider using a much larger value to reduce overall gas consumption.

- (new) Reading the owner's balance in the loop of `getTokensByOwner` function in **AzuroBet** contract consumes gas at each iteration. Declare a local variable with this value before the loop.

Project management (fixed)

The **.gitignore** file contains `package-lock.json` file. We recommend removing if from **.gitignore** and responsibly updating it.
Update 1: The issue has been fixed and is not present in the latest version of the code.

Notes

Large bet processing

Larger bets change odds and cause slippage. Larger bets will give gamblers an unfair advantage and increase risks for the bookmaker and liquidity providers if this effect is not addressed. The project splits large bets into smaller chunks (each up to 1% of the current pool) and aggressively reduces odds after the first one. It produces undesirable side-effects, e.g., placing five 1% bets is more effective than placing one 5%, and two 0.5% bets lose to a single 1%.

We recommend utilizing a continuous function and integrating it to get the exact odds for the particular bet. This idea is inspired by bonding curves of AMM protocols.

No multi-choice support

All conditions in the system have two outcomes and are independent of each other. Some events cannot be properly described as a combination of these binary conditions.

This analysis was performed by Pessimistic:

Daria Korepanova, Security Engineer
Evgeny Marchenko, Senior Security Engineer
Nikita Kirillov, Junior Security Engineer
Ivan Gladkikh, Junior Security Engineer
Boris Nikashin, Analyst
Irina Vikhareva, Project Manager

April 29, 2022