azuro × PESSIMISTIC

SECURITY ANALYSIS

by Pessimistic

This report is public

March 31, 2025

ABSTRACT

In this report, we consider the security of smart contracts of Azuro V3 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 Azuro V3 smart contracts. The audit was the recheck of the update made between the second and the third versions of the Azuro project. We described the audit process and provided details on the Project description in the sections below.

The audit showed several issues of medium severity: Documentation, Incorrect calculation of locked liquidity, Not all bets are accounted during rejection, Project's owner role, and Mismatch between contracts and provided repositories. Also, several low-severity issues were found.

The critical issue The ability of users deposits withdrawal was discovered by other auditors.

After the initial audit, the codebase was updated. The developers increased the number of tests and code coverage. All issues were fixed or commented.

GENERAL RECOMMENDATIONS

We have no any further recommendations.

PROJECT OVERVIEW

Project description

For this audit, we were provided with the Azuro V3 project in a private GitHub repository, branch `v3.0.4`, and commit 74cfe81c5f04934ca2ec4c2f37642884b28f86ed. Additionally, we were provided with two public repositories, which were used in the code:

- LiquidityTree on commit f83c7e55e26bf1171ad25d645ec7c795b13be7b6;
- Access on commit 98fcf56856135b8df6db071f82a4e3082e4a4731.

This is the third version of the project. Previously, we audited version two. This report does not include a full audit but focuses on reviewing the updates between version two and version three of the code.

The previous version of the code was divided into several parts, which were audited separately. Links to the reports can be found below. We have attached only those reports that have an impact on the current scope.

- "Azuro V2 Security Analysis by Pessimistic", last recheck was on January, 2024;
- "LiveCore Security Analysis by Pessimistic", last recheck was on January, 2024;
- "BetExpress Security Analysis by Pessimistic", last recheck was on March, 2024;
- "CashOut Security Analysis by Pessimistic", last recheck was on October, 2024.

The scope of the current difference review included:

- **contracts/extensions/Cashout.sol**;
- **contracts/extensions/Relayer.sol**;
- **contracts/utils/LiquidityTree.sol** (was copied from the LiquidityTree repository);
- **contracts/utils/OrderTools.sol**;
- **contracts/Factory.sol**;
- **contracts/AzuroBet.sol**;
- **contracts/Access.sol** (was copied from the Access repository);
- **contracts/LP.sol**;
- **contracts/Vault.sol**;
- **contracts/LiveCore.sol**.

The documentation for the update was not provided.

The tests and code coverage results:

- Azuro V3 repository: 119 out of 119 tests passed successfully, and the code coverage was 83.98%;
- LiquidityTree repository: 69 out of 69 tests passed successfully, and the code coverage was 96.73% ;
- Access repository: 15 out of 15 tests passed successfully, and the code coverage was 53.03%.

The total LOC of the project code is 3172.

Codebase update #1

After the initial audit, the codebase was updated. For the recheck, we were provided with commit 5960613123a288cf7033c300ce173960129b9da4 of **Azuro V3** repository.

This update included fixes or comments for all issues. The number of tests of **Azuro V3** repository increased. All 122 tests passed. The code coverage increased to 84.41%.

AUDIT PROCESS

We conducted this differences review on March 17-20, 2025.

This is the third version of the code, and this review includes the updates from the second version. Information about previous reports, provided repositories, and scope can be found in the Project description section.

We inspected the materials provided for the audit.

During the process, 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:

- The extent of the owner's rights;
- Whether the value of locked liquidity is calculated correctly in ordinary and combo bets;
- Whether the **LiquidityTree** and **Access** contracts logic is the same as in the provided repositories;
- Whether the signatures are handled correctly;
- Whether the update does not break the logic which was in the previous version of the code;
- Whether rejected bets are accounted for properly;
- Standard Solidity issues.

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 ran tests and calculated the code coverage for the 3 provided repositories.

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

From March 28 to March 31, we made the recheck for the current scope.

We analyzed the update and whether the issues were fixed, re-ran the tests and re-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. The ability of users deposits withdrawal (Found by other auditors, fixed)

Anyone can call the `LP.withdrawLiquidityFor` function. This allows anyone to withdraw any user's deposit from the **Vault**.

> 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.

M01. Documentation (commented)

The update has no documentation. 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.

> `Comment from the developers:`
> **The protocol version 3 reduced complexness (in compare with version 2) and actually most documentation is bet parameters diagram.**

M02. Incorrect calculation of locked liquidity (fixed)

The `LiveCore._rejectConditionBets` function rejects ordinary bets, updates payouts and locks liquidity. The `_calcReserve` function (is called inside the `_rejectConditionBets` function) has incorrect passed `payout` parameter as it is equal to the sum of rejected bets' payouts. However, it should be equal to the updated payout from the condition (`condition.payouts`, which is reduced by the sum of rejected bets' payouts) instead.

This leads to the incorrect accounting of locked liquidity.

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

M03. Not all bets are accounted during rejection (fixed)

The recalculation of the combined betting odds in the `LiveCore.rejectComboBets` function excludes unresolved bets that should not be rejected.

This happens since `condition.settledAt = 0` (as condition is not resolved yet) and `bet.timestamp` is more than 0. It leads to incorrect calculation of bet payout and an incorrect value of locked liquidity.

Consider changing the check at line 140 to the `condition.settledAt != 0 & bet.timestamp >= condition.settledAt` in the `rejectComboBets` to consider all bets.

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

M04. **Project owner's role** (commented)

In the current implementation, the system depends heavily on the owner role. Thus, there are scenarios that can lead to undesirable consequences for the project and its users, e.g., if owner's private keys become compromised. The owner of the project can:

- Change the **Relayer** address, affiliate address, fee, minimum deposit value, withdrawal timeout, the data provider address, the **LP** address of the previous protocol version and update core settings;
- Change the core beacons in the **Factory** contract;
- Change the admin address in the **Vault** contract which can withdraw liquidity, lock/unlock liquidity;
- Change the transferability of access tokens, burn access tokens, grant roles for rejecting orders, cancel conditions, allow oracles to place orders, and modify role settings in the **Access** contract.

Also, oracle may not sign messages for cashout requests, or may only sign if bet payout is already paid or combo bet is partially resolved. It would lead to revert in the `CashOut.cashOutBets` function. Furthermore, the oracle may refuse to sign messages, censoring or delaying bet placements in the `LiveCore.putOrder` function.

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

> `Comment from the developers:`
>
> We are using multisig for owner account.

M05. Mismatch between contracts and provided repositories (fixed)

The **LiquidityTree** and **Access** contracts are used in the **Azuro V3** repository and copied to this repository from the LiquidityTree and Access repositories.

According to the developers' comments, they should be identical. However, there are minor differences between them. These differences do not affect the project in the **Access** contract but do affect the **LiquidityTree** contract in the following functions: `_removeLimit`, `_nodewithdraw`.

These functions have `internal` visibility in the repository and `public` visibility in the **LiquidityTree** repository. It means that anyone can change leaves values inside the liquidity tree, which is used to store users liquidity.

Consider ensuring that the visibility settings are consistent between the copied contract and the **LiquidityTree** repository.

> `Comment from the developers:`
>
> Fixed function visibility at **LiquidityTree** repository for version v0.0.6, commit 5a9adede53866ff8217e7884333b501e3150f6eb.

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. Incorrect indexing (fixed)

In the `for`-loop at line 351 of the `Access._bytes32ToString` function, the counter `i` is not initialized to `0`. This causes non-zero bytes to be skipped and results in copying zero bytes to the `bytesArray` at line 352.

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

L02. Misleading comment (fixed)

The NatSpec comment at line 180 of the `CashOut.cashOutBets` function does not match the actual code (`isValidSignatureNow` function at line 209). In the code, the `betOwner` is either equal to the signer or equal to the contract address from EIP-1271.

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

L03. Revert if bet not found (fixed)

Consider adding a `revert` in the `LiveCore._resetSubBetOdds` function after the inner loop if nothing is found in the cycle. This would indicate that the desired outcome/condition pair was not found, which might be not expected.

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

L04. Unused functions (commented)

The `_add` and `_remove` functions of the **LiquidityTree** contract are not used in the current version of the code.

> `Comment from the developers:`
>
> **LiquidityTree used in the project as library contract, and used as is.**

Notes

N01. Signed and passed data is not checked (commented)

The `LiveCore.putOrder` function does not validate whether the `data` (signed by bettor) and the `order` (passed by relayer) parameters match.

It means that the **LiveCore** contract must trust how the **Relayer** assembles the `structHash` and `messageHash`. For instance, if the **Relayer** is malicious, they may pass an incorrect `betOwner` address to the `LiveCore.putOrder` function and receive a new minted **AzuroBet** token instead of the real bettor.

> `Comment from the developers:`
>
> **Relayer and LiveCore contracts owned by one owner and are at same security level. Whole order data are verified and signed by the oracle.**

N02. The canceled condition is handled differently during rejections (commented)

The rejection of ordinary bets (`rejectBets`) and combo bets (`rejectComboBets`) functions handle the canceled condition status (`ConditionState.CANCELED`) differently in the **LiveCore** contract:

- The `rejectBets` function reverts if the condition status is canceled or resolved in the `_checkConditionNotSettled` function;
- The `rejectComboBets` function reverts in case of resolved status in the `_resetSubBetOdds` function but continues execution if the status is canceled at line 139.

> `Comment from the developers:`
>
> **Difference is because of `rejectBets()` trying reject ordinar bet and the bet is completely depends on the condition state. For successfull bet rejection it (the condition) must be actual, not settled nor canceled. So if the condition resolved/canceled it can't be rejected (changed) so it is reverts.**
>
> **`rejectComboBets()` deals with set of subbets of different conditions. At time of rejecting if subbet's conditions are settled (or canceled) - its just ignored at odds calculation. Whole combo can`t be reverted because of one component is canceled.**

This analysis was performed by Pessimistic:

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

March 31, 2025