Azuro Security Review

Pashov Audit Group

Conducted by: pashov

October 30th, 2023

Contents

1. About pashov

Krum Pashov, or **pashov**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Check his previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **Azuro** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Azuro

Copied from the previous security reviews

Azuro is a decentralized betting protocol. Anyone can launch a frontend service that connects to the smart contracts and to receive an affiliate bonus for each bet made through the given frontend. Different betting events can be hosted, for example a football game. Odds are provided once by a Data Feed provider (Oracle) for initialization and then odds change based on the betting on the platform. A user bet gets automatically converted to an NFT in the user's wallet.

Continued

The `SuperExpress` functionality allows for betting for the outcomes of previously fixed 15 events. Bets are accepted until a round starts. A bet maker can choose one of three possible outcomes for each of the 15 events in the round. If at least 9 event outcomes are correctly guessed, then the bet is a winning bet and gets a proportional prize from the prize fund. If at least 14 event outcomes are correctly guessed in a bet then it will also receive a portion of the super prize fund which is in the form of `AZUR` tokens. A round is called a "super round" whenever a part of the super prize fund is paid out. While the prize fund is sourced from the bets in a round, the super prize fund comes from liquidity providers (the logic is `LP` contract).

Observations

Both the `SuperExpress` and `LP` contracts are upgradeable, which poses a centralization risk to the protocol and its users. The `SuperExpress` admin can cancel any active round and change winning outcomes, so he has total control over the bets made on the platform. Also each of the LP, front-end or the DAO fee can be set to 100%, or all three can add up to 100%.

Threat Model

Privileged Roles & Actors

- LP's SuperExpress - the only account allowed to use any of `LP`'s functionality
- LP admin - can set the `superExpress` account address
- SuperExpress admin - has total control over betting rounds, can add rounds, cancel them, complete them, set the winning outcomes and set the contract's fees & `betNominal`
- Super prize liquidity provider - deposits Azuro tokens in `SuperExpress` for the super prize fund and can claim fees from each bet
- Bettor - can place bets (one or multiple, for himself or for another account)
- Front-end referral - can claim a referral fee from the bets made through it in `SuperExpress`
- DAO - can withdraw the `collectedDaoFee` from the bets in `SuperExpress`

Security Interview

Q: What in the protocol has value in the market?

A: The prize & super prize funds have economical value.

Q: In what case can the protocol/users lose money?

A: Impossibility to withdraw their bet winning payouts or super prize liquidity provided.

Q: What are some ways that an attacker achieves his goals?

A: Exploit reward payouts, fees or LP payouts calculations.

5. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hash **- 3369c3ea9e62897046644cf5dcb845e0e8d61766b**

Scope

The following smart contracts were in scope of the audit:

- `SuperExpress`
- `LP`
- `libraries/FixedMath`

7. Executive Summary

Over the course of the security review, pashov engaged with Azuro to review Azuro. In this period of time a total of **5** issues were uncovered.

Protocol Summary

Protocol Name	Azuro
Date	October 30th, 2023

Findings Count

Severity	**Amount**
Critical	1
Medium	1
Low	3
Total Findings	**5**

Summary of Findings

ID	Title	Severity	Status
[C-01]	Front-end owner can steal all bets balance	Critical	Resolved
[M-01]	Sum of fee percentages might not be equal to the admin provided total value	Medium	Resolved
[L-01]	Division before multiplication using FixedMath library	Low	Resolved
[L-02]	Betters might not have time to bet in a round	Low	Resolved
[L-03]	Re-assignment after setting a value to a local variable	Low	Resolved

8. Findings

8.1. Critical Findings

[C-01] Front-end owner can steal all bets balance

Severity

Impact: High, as it results in 100% loss for the protocol & its users

Likelihood: High, as any Azuro SuperExpress front-end owner can exploit this

Description

The Azuro protocol allows any front-end owner to add his address in each bet placed through his UI. In `SuperExpress`, this address will be able to withdraw a fee amount that is taken from each bet, some small percentage. The problem is that the `withdrawFrontFee` method, from where this happens, is flawed - it can be called multiple times in a loop until there is no more bet token balance in the contract.

The flaw is that even though the method does this:

```
round.frontRewarded[msg.sender] = true;
```

which is an attempt at stopping the front-end owner from claiming his rewards again, this is not checked anywhere. The `_viewFrontFee` method should account for already claimed "front" rewards, but it doesn't.

Recommendations

In `_viewFrontFee` check if `round.frontRewarded[msg.sender] == true` and if it is just return 0.

8.2. Medium Findings

[M-01] Sum of fee percentages might not be equal to the admin provided `total` value

Severity

Impact: High, as it will mess the accounting of the contract, resulting in stuck/lost funds

Likelihood: Low, as it requires a mistake or an error on the admin side

Description

The input validation for the `newFeePercents` in the `_changeFee` method looks like this:

```
if (
            (newFeePercents.lp + newFeePercents.front + newFeePercents.dao) >
            FixedMath.ONE ||
            newFeePercents.total > FixedMath.ONE
        ) revert IncorrectValue();
```

While both the sum of the fee percentages and the total value are validated that they are not more than 100%, it is possible that the sum of the three different fee percentages is not equal to the `total` value. Now if they actually amount to different values this will mess up the accounting of the contract, as when a user puts a bet then the `fee.total` is removed from his bet amount, but then different values would be sent to the fee recipients. This will result in either stuck funds or direct value loss for the protocol & its users.

Recommendations

Change the code in the following way:

```
if (
-             (newFeePercents.lp + newFeePercents.front + newFeePercents.dao) >
-             FixedMath.ONE ||
+
+ (newFeePercents.lp + newFeePercents.front + newFeePercents.dao) != newFeePercents.to
            newFeePercents.total > FixedMath.ONE
        ) revert IncorrectValue();
```

8.3. Low Findings

[L-01] Division before multiplication using `FixedMath` library

The `withdrawView` method in `LP` has the following piece of code:

```
amount = shareAmount.mul(
       superRound_.totalPrizeFund.div(superRound_.totalShare)
);
```

The `div` method here is called before the `mul` method, which is division before multiplication. While it is currently not problematic, because of the way those methods work coming from the `FixedMath` library, it is still a convention and a best practice to always multiply before dividing in Solidity. Suggestion is to change code to

```
amount = (shareAmount.mul(superRound_.totalPrizeFund)).div
  (superRound_.totalShare);
```

[L-02] Betters might not have time to bet in a round

Currently, the `addRound` method allows setting `block.timestamp` as the `round.start` value. The problem is that in `_putBet` it is checked that if `block.timestamp >= round.start` then the transaction will revert. It is recommended that a minimum value of a period before a round starts is set (for example 1 day) so that users are guaranteed to be allowed to bet.

[L-03] Re-assignment after setting a value to a local variable

In `SuperExpress::resolveRound` there is this assignment

```
uint256 roundFeeLP = round.feeLP;
```

but a few lines below there is this one

```
roundFeeLP = compoundRoundFee_.mul(feePercents_.lp);
```

which will actually overwrite the first one. This shows potentially other intentions by the developer or is not needed. I suggest just declaring `roundFeeLP` without giving it a value, as this should be the correct implementation.