4th05

Zaros

Security Review Report

31 July 2024

Security Review Report

4th05

July 31, 2024

Table of Contents

Protocol Summary

Zaros is a Perpetuals DEX powered by Boosted (Re)Staking Vaults. It seeks to maximize LPs yield generation, while offering a top-notch trading experience on Arbitrum (and Monad in the future).

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.

Risk Classification

		Impact		
		High	Medium	Low
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

Overview

Contest platform	Codehawks
LOC	2878
Language	Solidity
Commit	d687fe96bb7ace8652778797052a38763fbcbb1b
Previous audits	Cyfrin audit

Scope

- src/utils/Math.sol
- src/utils/Constants.sol
- src/utils/Errors.sol
- src/tree-proxy/leaves/RootUpgrade.sol
- src/tree-proxy/leaves/Branch.sol
- src/tree-proxy/leaves/LookupTable.sol
- src/tree-proxy/RootProxy.sol
- src/tree-proxy/branches/UpgradeBranch.sol
- src/tree-proxy/branches/LookupBranch.sol
- src/external/chainlink/ChainlinkUtil.sol
- src/external/chainlink/keepers/BaseKeeper.sol
- src/external/chainlink/interfaces/ILogAutomation.sol
- src/external/chainlink/interfaces/IStreamsLookupCompatible.sol
- src/external/chainlink/interfaces/IAutomationCompatible.sol
- src/external/chainlink/keepers/market-order/MarketOrderKeeper.sol
- src/account-nft/AccountNFT.sol
- src/external/chainlink/interfaces/IFeeManager.sol
- src/external/chainlink/interfaces/IOffchainAggregator.sol
- src/external/chainlink/interfaces/IVerifierProxy.sol
- src/external/chainlink/interfaces/IAggregatorV3.sol
- src/external/chainlink/keepers/liquidation/LiquidationKeeper.sol
- src/usd/USDToken.sol
- src/perpetuals/PerpsEngine.sol
- src/account-nft/interfaces/IAccountNFT.sol
- src/perpetuals/branches/TradingAccountBranch.sol
- src/perpetuals/branches/LiquidationBranch.sol
- src/perpetuals/branches/OrderBranch.sol
- src/perpetuals/branches/PerpMarketBranch.sol
- src/perpetuals/leaves/OrderFees.sol
- src/perpetuals/leaves/CustomReferralConfiguration.sol
- src/perpetuals/branches/SettlementBranch.sol
- src/perpetuals/branches/GlobalConfigurationBranch.sol
- src/perpetuals/leaves/MarginCollateralConfiguration.sol
- src/perpetuals/leaves/GlobalConfiguration.sol
- src/perpetuals/leaves/PerpMarket.sol
- src/perpetuals/leaves/Position.sol

- src/perpetuals/leaves/SettlementConfiguration.sol
- src/perpetuals/leaves/MarketConfiguration.sol
- src/perpetuals/leaves/TradingAccount.sol
- src/perpetuals/leaves/Referral.sol
- src/perpetuals/leaves/MarketOrder.sol
- src/perpetuals/leaves/FeeRecipients.sol
- src/perpetuals/leaves/OffchainOrder.sol

Issues found

Severity	Number of issues found
High	0
Medium	1
Low	0
Info	0

Findings

[M1] `GlobalConfigurationBranch::updatePerpMarketConfiguration` missing update of the `priceFeedHeartbeatSeconds` value

Summary

In the `GlobalConfigurationBranch.sol`, calling the `updatePerpMarketConfiguration` function, the `priceFeedHeartbeatSeconds` set in the `MarketConfiguration.sol` cannot be updated as all the other parameters.

Vulnerability Details

In the `GlobalConfigurationBranch.sol::updatePerpMarketConfiguration` the `priceFeedHeartbeatSeconds` input value does not get updated because it is not assigned to the right `uint 32 priceFeedHeartbeatSeconds` variable in the `MarketConfiguration.sol::update`.

That input it is not assigned to any variable at all.

```
1      function updatePerpMarketConfiguration(
2          uint128 marketId,
3          UpdatePerpMarketConfigurationParams calldata params
4      )
5          external
6          onlyOwner
7          onlyWhenPerpMarketIsInitialized(marketId)
8      {
9          PerpMarket.Data storage perpMarket = PerpMarket.load(marketId);
10         MarketConfiguration.Data storage perpMarketConfiguration =
               perpMarket.configuration;
11
12         if (abi.encodePacked(params.name).length == 0) {
13             revert Errors.ZeroInput("name");
14         }
15         if (abi.encodePacked(params.symbol).length == 0) {
16             revert Errors.ZeroInput("symbol");
17         }
18         if (params.priceAdapter == address(0)) {
19             revert Errors.ZeroInput("priceAdapter");
20         }
21         if (params.maintenanceMarginRateX18 == 0) {
22             revert Errors.ZeroInput("maintenanceMarginRateX18");
23         }
24         if (params.maxOpenInterest == 0) {
25             revert Errors.ZeroInput("maxOpenInterest");
26         }
27         if (params.maxSkew == 0) {
28             revert Errors.ZeroInput("maxSkew");
29         }
30         if (params.initialMarginRateX18 == 0) {
31             revert Errors.ZeroInput("initialMarginRateX18");
32         }
33         if (params.initialMarginRateX18 <= params.
               maintenanceMarginRateX18) {
34             revert Errors.
                   InitialMarginRateLessOrEqualThanMaintenanceMarginRate();
35         }
36         if (params.skewScale == 0) {
37             revert Errors.ZeroInput("skewScale");
38         }
39         if (params.minTradeSizeX18 == 0) {
40             revert Errors.ZeroInput("minTradeSizeX18");
41         }
42         if (params.maxFundingVelocity == 0) {
43             revert Errors.ZeroInput("maxFundingVelocity");
44         }
45 @>      if (params.priceFeedHeartbeatSeconds == 0) {
46             revert Errors.ZeroInput("priceFeedHeartbeatSeconds");
47         }
```

```
48
49    @>      perpMarketConfiguration.update(
50              MarketConfiguration.Data({
51                  name: params.name,
52                  symbol: params.symbol,
53                  priceAdapter: params.priceAdapter,
54                  initialMarginRateX18: params.initialMarginRateX18,
55                  maintenanceMarginRateX18: params.
                        maintenanceMarginRateX18,
56                  maxOpenInterest: params.maxOpenInterest,
57                  maxSkew: params.maxSkew,
58                  maxFundingVelocity: params.maxFundingVelocity,
59                  minTradeSizeX18: params.minTradeSizeX18,
60                  skewScale: params.skewScale,
61                  orderFees: params.orderFees,
62    @>            priceFeedHeartbeatSeconds: params.
          priceFeedHeartbeatSeconds
63              })
64          );
65
66          emit LogUpdatePerpMarketConfiguration(msg.sender, marketId);
67      }
```

Impact

The owner is not able to update the `priceFeedHeartbeatSeconds` parameter in case of high market volatility or lower available heartbeat of all the `oracles`, accepting this way outdated prices for the market

Tools Used

Manual review

Recommendations

Add a line of code in the `MarketConfiguration.sol::update` to update the `priceFeedHeartbeatSecond`

```
1  function update(Data storage self, Data memory params) internal {
2
3  self.name = params.name;
4
5  self.symbol = params.symbol;
6
7  self.priceAdapter = params.priceAdapter;
```

```
 8
 9  self.initialMarginRateX18 = params.initialMarginRateX18;
10
11  self.maintenanceMarginRateX18 = params.maintenanceMarginRateX18;
12
13  self.maxOpenInterest = params.maxOpenInterest;
14
15  self.maxSkew = params.maxSkew;
16
17  self.maxFundingVelocity = params.maxFundingVelocity;
18
19  self.minTradeSizeX18 = params.minTradeSizeX18;
20
21  self.skewScale = params.skewScale;
22
23  self.orderFees = params.orderFees;
24
25  + self.priceFeedHeartbeatSeconds = params.priceFeedHeartbeatSeconds
26
27  }
```