ZooDAO Arbitrum Battles Audit Report

Jan 2, 2024

Table of Contents

Summary

This report has been prepared for ZooDAO smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

Project Summary

Project Name	**ZooDAO**
Codebase	**https://github.com/ZooDAO-Project/arbitrum-battles**
Commit	**563e14ad0ccf7b1a2eb5838c79b3ec6b30f30a27**
Language	**Solidity**

Audit Summary

Delivery Date	**Jan 2, 2024**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**13**

[WP-H1] `votingPosition.lastEpochOfIncentiveReward` can be incorrectly overwritten, causing the voter to receive incentive rewards that do not belong to them.

High

Issue Description

When `addDaiToVoting()` is called at stage > 2, `lastEpochOfIncentiveReward` will be set to `currentEpoch + 1` instead of `currentEpoch` to avoid the votingPosition from receiving incentive rewards for the epoch they didn't stay in.

However, if the user calls `computeIncentiveRewardForVoter()`, `votingPosition.lastEpochOfIncentiveReward` will be overwritten to the epoch when the voter entered, resulting in the voter receiving incentive rewards that do not belong to them.

https://github.com/ZooDAO-Project/arbitrum-battles/blob/b32057492ffa92fc04fdd954e4d746134c539270/contracts/NftBattleArena.sol#L1211-L1233

```
1211      function computeInvenctiveRewardForVoter(uint256 votingPositionId) internal
      returns (uint256 reward)
1212      {
1213          VotingPosition storage votingPosition =
      votingPositionsValues[votingPositionId];
1214          uint256 stakingPositionId = votingPosition.stakingPositionId;
1215
1216          address collection = stakingPositionsValues[stakingPositionId].collection;
1217          updateInfo(stakingPositionId);
1218          updateInfoAboutStakedNumber(collection);
      // Updates info about collection.
1219
1220          uint256 lastEpoch = computeLastEpoch(votingPositionId); // Last epoch
1221          if (lastEpoch > endEpochOfIncentiveRewards)
1222              lastEpoch = endEpochOfIncentiveRewards;
1223          if (pendingVotesEpoch[votingPositionId] != 0 && lastEpoch >
      pendingVotesEpoch[votingPositionId])
1224              lastEpoch = pendingVotesEpoch[votingPositionId];
1225
1226          for (uint256 i = votingPosition.lastEpochOfIncentiveReward; i < lastEpoch;
      ++i)
```

```
1227          {
1228              if (poolWeight[address(0)][i] != 0 &&
      rewardsForEpoch[stakingPositionId][i].isWinnerChose) // Check that collection has
      non-zero weight in veZoo and nft played in battle.
1229                  reward += baseVoterReward * votingPosition.daiVotes *
      poolWeight[collection][i] / (poolWeight[address(0)][i] * playedVotesByEpoch[i]);
1230          }
1231
1232          votingPosition.lastEpochOfIncentiveReward = lastEpoch;
1233      }
```

Recommendation

Return earlier when `votingPosition.lastEpochOfIncentiveReward > lastEpoch` :

```
1211      function computeInvenctiveRewardForVoter(uint256 votingPositionId) internal
      returns (uint256 reward)
1212      {
1213          VotingPosition storage votingPosition =
      votingPositionsValues[votingPositionId];
1214          uint256 stakingPositionId = votingPosition.stakingPositionId;
1215
1216          address collection = stakingPositionsValues[stakingPositionId].collection;
1217          updateInfo(stakingPositionId);
1218          updateInfoAboutStakedNumber(collection);
      // Updates info about collection.
1219
1220          uint256 lastEpoch = computeLastEpoch(votingPositionId); // Last epoch
1221          if (votingPosition.lastEpochOfIncentiveReward > lastEpoch) {
1222              return 0;
1223          }
1224          if (lastEpoch > endEpochOfIncentiveRewards)
1225              lastEpoch = endEpochOfIncentiveRewards;
1226          if (pendingVotesEpoch[votingPositionId] != 0 && lastEpoch >
      pendingVotesEpoch[votingPositionId])
1227              lastEpoch = pendingVotesEpoch[votingPositionId];
1228
1229          for (uint256 i = votingPosition.lastEpochOfIncentiveReward; i < lastEpoch;
      ++i)
1230          {
1231              if (poolWeight[address(0)][i] != 0 &&
      rewardsForEpoch[stakingPositionId][i].isWinnerChose) // Check that collection has
      non-zero weight in veZoo and nft played in battle.
```

```
1232                    reward += baseVoterReward * votingPosition.daiVotes *
      poolWeight[collection][i] / (poolWeight[address(0)][i] * playedVotesByEpoch[i]);
1233            }
1234
1235            votingPosition.lastEpochOfIncentiveReward = lastEpoch;
1236        }
```

Status

✓ Fixed

[WP-M2] Wrong calculation of `lpZooDiff` will cause `withdrawDaiFromVotingPositionStablecoin()` to always revert.

Medium

Issue Description

`nftBattleArena.withdrawDaiFromVoting()` will withdraw ZooLP, therefore, it should be `afterBalance - beforeBalance` .

The current implementation will most certainly result in a revert.

https://github.com/ZooDAO-Project/arbitrum-battles/blob/b32057492ffa92fc04fdd954e4d746134c539270/contracts/NftVotingPosition.sol#L138-L160

```
138     function withdrawDaiFromVotingPositionStablecoin(uint256 votingPositionId,
    address beneficiary, uint256 daiNumber, uint256 minOut, address tokenToReceive)
    external onlyVotingOwner(votingPositionId)
139     {
140         uint256 balanceBeforeWithdraw = dai.balanceOf(address(this));
141         uint256 lpZooBalanceBeforeWithdraw = lpZoo.balanceOf(address(this));
142 
143         nftBattleArena.withdrawDaiFromVoting(votingPositionId, address(this),
    address(this), daiNumber, false);
144         uint256 diff = dai.balanceOf(address(this)) - balanceBeforeWithdraw;
145 
146         // unstake tokens and receive in tokenToReceive
147         uint256 amountOut = glpRewardRouter.unstakeAndRedeemGlp(tokenToReceive,
    diff, minOut, address(this));
148 
149         // take fee 1.5% of tokenToReceive
150         IERC20(tokenToReceive).transfer(team, 15 * amountOut / 1000);
151 
152         // transfer 98.5% of tokenToReceive to msg.sender
153         IERC20(tokenToReceive).transfer(msg.sender, 985 * amountOut / 1000);
154 
155         // Send lpZoo to beneficiary if needed
156         uint256 lpZooDiff = lpZooBalanceBeforeWithdraw -
    lpZoo.balanceOf(address(this));
157         if (lpZooDiff != 0) {
158             lpZoo.transfer(beneficiary, lpZooDiff);
```

```
159         }
160     }
```

Recommendation

```
138     function withdrawDaiFromVotingPositionStablecoin(uint256 votingPositionId,
address beneficiary, uint256 daiNumber, uint256 minOut, address tokenToReceive)
external onlyVotingOwner(votingPositionId)
139     {
140         uint256 balanceBeforeWithdraw = dai.balanceOf(address(this));
141         uint256 lpZooBalanceBeforeWithdraw = lpZoo.balanceOf(address(this));
142 
143         nftBattleArena.withdrawDaiFromVoting(votingPositionId, address(this),
address(this), daiNumber, false);
144         uint256 diff = dai.balanceOf(address(this)) - balanceBeforeWithdraw;
145 
146         // unstake tokens and receive in tokenToReceive
147         uint256 amountOut = glpRewardRouter.unstakeAndRedeemGlp(tokenToReceive,
diff, minOut, address(this));
148 
149         // take fee 1.5% of tokenToReceive
150         IERC20(tokenToReceive).transfer(team, 15 * amountOut / 1000);
151 
152         // transfer 98.5% of tokenToReceive to msg.sender
153         IERC20(tokenToReceive).transfer(msg.sender, 985 * amountOut / 1000);
154 
155         // Send lpZoo to beneficiary if needed
156         uint256 lpZooDiff = lpZoo.balanceOf(address(this)) -
lpZooBalanceBeforeWithdraw;
157         if (lpZooDiff != 0) {
158             lpZoo.transfer(beneficiary, lpZooDiff);
159         }
160     }
```

Status

✓ Fixed

[WP-G3] Unnecessary max approve

Gas

Issue Description

https://github.com/ZooDAO-Project/arbitrum-battles/blob/b32057492ffa92fc04fdd954e4d746134c539270/contracts/NftBattleArena.sol#L364-L375

```
364     function createVotingPosition(uint256 stakingPositionId, address voter,
    uint256 amount) external only(nftVotingPosition) returns (uint256 votes, uint256
    votingPositionId)
365     {
366         //require(getCurrentStage() == Stage.SecondStage, "Wrong stage!"); //
    Require turned off cause its moved to voting position contract due to lack of
    space for bytecode. // Requires to be at second stage of battle epoch.
367
368         updateInfo(stakingPositionId);
    // Updates staking position params from previous epochs.
369
370         dai.approve(address(vault), type(uint256).max);
    // Approves Dai for yearn.
371         uint256 yTokensNumber = vault.balanceOf(address(this));
372         require(vault.mint(amount) == 0);
    // Deposits dai to yearn vault and get yTokens.
373
374         (votes, votingPositionId) = _createVotingPosition(stakingPositionId,
    voter, vault.balanceOf(address(this)) - yTokensNumber, amount);// Calls internal
    create voting position.
375     }
```

`dai.approve(address(vault), type(uint256).max);` can be done in the constructor function instead of each time `createVotingPosition()` is called.

Status

✓ Fixed

[WP-G4] Using immutable variable can save gas

Gas

Issue Description

Considering that the following variables will never change, changing them to immutable variables instead of storage variables can save gas.

`NftBattleArena.zooVoteRateNominator` , `NftBattleArena.zooVoteRateDenominator`

https://github.com/ZooDAO-Project/arbitrum-battles/blob/b32057492ffa92fc04fdd954e4d746134c539270/contracts/NftBattleArena.sol#L252-L262

```
252     /// @param _zooVoteRateNominator - amount of votes for 1 LP with zoo.
253     /// @param _zooVoteRateDenomibator - divider for amount of votes for 1 LP with
    zoo.
254     /// @param _zoo actual zoo token(not LP).
255     function init(uint256 _zooVoteRateNominator, uint256 _zooVoteRateDenomibator,
    IERC20Metadata _zoo) external
256     {
257         require(zooVoteRateNominator == 0);
258 
259         zooVoteRateNominator = _zooVoteRateNominator;
260         zooVoteRateDenominator = _zooVoteRateDenomibator;
261         zoo = _zoo;
262     }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/b32057492ffa92fc04fdd954e4d746134c539270/contracts/NftBattleArena.sol#L165-L166

```
165     uint256 public zooVoteRateNominator; // amount of votes for 1 LP with zoo.
166     uint256 public zooVoteRateDenominator;
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/b32057492ffa92fc04fdd954e4d746134c539270/contracts/NftBattleArena.sol#L217-L250

```
217     /// @notice Contract constructor.
218     /// @param _lpZoo - address of LP token with zoo.
219     /// @param _dai - address of stable token contract.
220     /// @param _vault - address of yearn.
221     /// @param _zooGovernance - address of ZooDao Governance contract.
222     /// @param _treasuryPool - address of ZooDao treasury pool.
223     ///  _teamAddress - address of ZooDao team reward pool.
224     constructor (
225         ERC4626 _lpZoo,
226         IERC20Metadata _dai,
227         address _vault,
228         address _zooGovernance,
229         address _treasuryPool,
230         // address _teamAddress,
231         address _nftStakingPosition,
232         address _nftVotingPosition,
233         address _veZoo)
234     {
235         lpZoo = _lpZoo;
236         dai = _dai;
237         vault = VaultAPI(_vault);
238         zooGovernance = ZooGovernance(_zooGovernance);
239         zooFunctions = IZooFunctions(zooGovernance.zooFunctions());
240         veZoo = ListingList(_veZoo);
241 
242         treasury = _treasuryPool;
243         // team = _teamAddress;
244         nftStakingPosition = _nftStakingPosition;
245         nftVotingPosition = _nftVotingPosition;
246 
247         epochStartDate = block.timestamp; // Start date of 1st battle.
248         epochsStarts[currentEpoch] = block.timestamp;
249         (firstStageDuration, secondStageDuration, thirdStageDuration,
    fourthStageDuration, fifthStageDuration, epochDuration) =
    zooFunctions.getStageDurations();
250     }
```

Status

ⓘ Acknowledged

[WP-H5] `BattleRewards` calculated by `NftBattleArena._calculateBattleRewards()` are sometimes inflated, resulting in more winnings/losses than what should actually be, due to the incorrect maintenance of `loserRewards1.yTokens` when treasury wins.

High

Issue Description

In `NftBattleArena._calculateBattleRewards()` L1079-L1084, when `winner == 0` (treasury wins), `loserRewards1.yTokens` should be set as `loserRewards.yTokens - income` .

Currently, `loserRewards1.yTokens` (`rewardsForEpoch[loser][currentEpoch + 1].yTokens`) is not being set, causing the inflated `battleReward1.yTokens` used in the next round (e.g., during the next `pairNft()`), where a portion of it no longer belongs to voters but has been transferred to the `treasury` , resulting in inflated interest (i.e., BattleRewards) calculated for the next round.

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L1066-L1128

```
1066      /// @dev Contains calculation logic of battle rewards
1067      /// @param winner stakingPositionId of NFT that WON in battle
1068      /// @param loser stakingPositionId of NFT that LOST in battle
1069      function _calculateBattleRewards(uint256 winner, uint256 loser) internal
1070      {
1071          BattleRewardForEpoch storage winnerRewards =
      rewardsForEpoch[winner][currentEpoch];
1072          BattleRewardForEpoch storage loserRewards =
      rewardsForEpoch[loser][currentEpoch];
1073
1074          BattleRewardForEpoch storage winnerRewards1 =
      rewardsForEpoch[winner][currentEpoch + 1];
1075          BattleRewardForEpoch storage loserRewards1 =
      rewardsForEpoch[loser][currentEpoch + 1];
1076
1077          if (winner == 0 || loser == 0) // arena 50-50 case
```

```
1078        {
1079            if (winner == 0) { // Battle Arena won
1080                // Take yield
1081                loserRewards.isWinnerChose = true;
1082                uint256 income = loserRewards.yTokens -
    tokensToShares(loserRewards.tokensAtBattleStart);
1083                require(vault.redeem(income) == 0);
1084                _stablecoinTransfer(treasury, dai.balanceOf(address(this)));
1085            } else {
1086            // Grant Zoo
1087                winnerRewards.zooRewards +=
    zooFunctions.getLeagueZooRewards(winnerRewards.league);
1088                winnerRewards.isWinnerChose = true;
1089            }
1090            return;
1091        }
1092
1093        // Skip if price per share didn't change since pairing
1094        uint256 currentPps = vault.exchangeRateCurrent();
1095        if (winnerRewards.pricePerShareAtBattleStart == currentPps)
1096        {
1097            return;
1098        }
1099
1100        winnerRewards.pricePerShareCoef = currentPps *
    winnerRewards.pricePerShareAtBattleStart / (currentPps -
    winnerRewards.pricePerShareAtBattleStart);
1101        loserRewards.pricePerShareCoef = winnerRewards.pricePerShareCoef;
1102
1103        // Income = yTokens at battle end - yTokens at battle start
1104        uint256 income1 = winnerRewards.yTokens -
    tokensToShares(winnerRewards.tokensAtBattleStart);
1105        uint256 income2 = loserRewards.yTokens -
    tokensToShares(loserRewards.tokensAtBattleStart);
1106
1107        require(vault.redeem(((income1 + income2) / 25)) == 0);           //
    Withdraws dai from vault for yTokens, minus staker %.
1108
1109        uint256 daiReward = dai.balanceOf(address(this));
1110        _stablecoinTransfer(treasury, daiReward);
    // Transfers treasury part. 4 / 100 == 4%
1111
1112        winnerRewards.yTokensSaldo += int256(((income1 + income2) * 96 / 100));
```

```
1113            loserRewards.yTokensSaldo -= int256(income2);
1114
1115            winnerRewards1.yTokens = winnerRewards.yTokens + income2 - ((income1 +
    income2) / 25);
1116            loserRewards1.yTokens = loserRewards.yTokens - income2; // Withdraw reward
    amount.
1117
1118            stakingPositionsValues[winner].lastUpdateEpoch = currentEpoch + 1;
    // Update lastUpdateEpoch to next epoch.
1119            stakingPositionsValues[loser].lastUpdateEpoch = currentEpoch + 1;
    // Update lastUpdateEpoch to next epoch.
1120            winnerRewards1.votes += winnerRewards.votes;
    // Update votes for next epoch.
1121            loserRewards1.votes += loserRewards.votes;
    // Update votes for next epoch.
1122
1123            winnerRewards1.league =
    zooFunctions.getNftLeague(winnerRewards1.votes);    // Update league for next
    epoch.
1124            loserRewards1.league =
    zooFunctions.getNftLeague(loserRewards1.votes);        // Update league for next
    epoch.
1125
1126            winnerRewards.isWinnerChose = true;
1127            loserRewards.isWinnerChose = true;
1128        }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L945-L1021

```
945     /// @notice Function for pair nft for battles.
946     /// @param stakingPositionId - id of staker position.
947     function pairNft(uint256 stakingPositionId) external
948     {
949         require(getCurrentStage() == Stage.ThirdStage, "Wrong stage!");
    // Requires to be at 3 stage of battle epoch.
950
951         updateInfo(stakingPositionId);
952         BattleRewardForEpoch storage battleReward1 =
    rewardsForEpoch[stakingPositionId][currentEpoch];
953
```

```
954         // this require makes impossible to pair if there are no available pair.
    // require(numberOfNftsWithNonZeroVotes / 2 > nftsInGame / 2, "E1");            //
    Requires enough nft for pairing.
955         uint256 index1;
    // Index of nft paired for.
956         uint256[] memory leagueList = new uint256[](numberOfNftsWithNonZeroVotes);
957         uint256 nftsInSameLeague = 0;
958         bool idFound;
959
960         // Find first staking position and get list of opponents from league for
    index2
961         for (uint256 i = nftsInGame; i < numberOfNftsWithNonZeroVotes; ++i)
962         {
963             updateInfo(activeStakerPositions[i]);
964             if (activeStakerPositions[i] == stakingPositionId)
965             {
966                 index1 = i;
967                 idFound = true;
968                 continue;
969                 // break;
970             }
971             // In the same league
972             else if (battleReward1.league ==
    rewardsForEpoch[activeStakerPositions[i]][currentEpoch].league)
973             {
974                 leagueList[nftsInSameLeague] = activeStakerPositions[i];
975                 nftsInSameLeague++;
976             }
977         }
978         require(idFound, "E1");
979
980         (activeStakerPositions[index1], activeStakerPositions[nftsInGame]) =
    (activeStakerPositions[nftsInGame], activeStakerPositions[index1]);// Swaps
    nftsInGame with index.
981         nftsInGame++;
    // Increases amount of paired nft.
982
983         uint256 stakingPosition2;
984         battleReward1.tokensAtBattleStart = sharesToTokens(battleReward1.yTokens);
    // Records amount of yTokens on the moment of pairing for candidate.
985         battleReward1.pricePerShareAtBattleStart = vault.exchangeRateCurrent();
986
987         if (nftsInSameLeague != 0)
```

```
988         {
989             uint256 index2;
990             stakingPosition2 = leagueList[0];
991             if (nftsInSameLeague > 1)
992             {
993                 stakingPosition2 = leagueList[zooFunctions.computePseudoRandom() %
    nftsInSameLeague];
994             }
995 
996             for (uint256 i = nftsInGame; i < numberOfNftsWithNonZeroVotes; ++i)
997             {
998                 if (activeStakerPositions[i] == stakingPosition2)
999                 {
1000                     index2 = i;
1001                 }
1002             }
1003 
1004             //updateInfo(stakingPosition2);
1005             BattleRewardForEpoch storage battleReward2 =
    rewardsForEpoch[stakingPosition2][currentEpoch];
1006             battleReward2.tokensAtBattleStart =
    sharesToTokens(battleReward2.yTokens);            // Records amount of yTokens on
    the moment of pairing for opponent.
1007             battleReward2.pricePerShareAtBattleStart =
    vault.exchangeRateCurrent();
1008 
1009             (activeStakerPositions[index2], activeStakerPositions[nftsInGame]) =
    (activeStakerPositions[nftsInGame], activeStakerPositions[index2]); // Swaps
    nftsInGame with index of opponent.
1010             nftsInGame++;
    // Increases amount of paired nft.
1011         }
1012         else
1013         {
1014             stakingPosition2 = 0;
1015         }
1016 
1017         pairsInEpoch[currentEpoch].push(NftPair(stakingPositionId,
    stakingPosition2, false, false));// Pushes nft pair to array of pairs.
1018         uint256 pairIndex = getNftPairLength(currentEpoch) - 1;
1019 
1020         emit PairedNft(currentEpoch, stakingPositionId, stakingPosition2,
    pairIndex);
```

```
1021      }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L1130-L1150

```
1130      /// @notice Function for updating position from lastUpdateEpoch, in case there was no battle with position for a while.
1131      function updateInfo(uint256 stakingPositionId) public
1132      {
1133          StakerPosition storage position = stakingPositionsValues[stakingPositionId];
1134          uint256 lastUpdateEpoch = position.lastUpdateEpoch; // Get lastUpdateEpoch for position.
1135          if (lastUpdateEpoch == currentEpoch) // If already updated in this epoch - skip.
1136              return;
1137
1138          for (; lastUpdateEpoch < currentEpoch; ++lastUpdateEpoch)
1139          {
1140              BattleRewardForEpoch storage rewardOfCurrentEpoch = rewardsForEpoch[stakingPositionId][lastUpdateEpoch + 1];
1141              BattleRewardForEpoch storage rewardOflastUpdateEpoch = rewardsForEpoch[stakingPositionId][lastUpdateEpoch];
1142
1143              rewardOfCurrentEpoch.votes += rewardOflastUpdateEpoch.votes; // Get votes from lastUpdateEpoch.
1144              rewardOfCurrentEpoch.yTokens += rewardOflastUpdateEpoch.yTokens; // Get yTokens from lastUpdateEpoch.
1145
1146              rewardOfCurrentEpoch.league = zooFunctions.getNftLeague(rewardOfCurrentEpoch.votes);
1147          }
1148
1149          position.lastUpdateEpoch = currentEpoch; // Set lastUpdateEpoch to currentEpoch.
1150      }
```

Status

✓ Fixed

[WP-M6] `updateInfoAboutStakedNumber(address(0))` may inflate the global poolWeight.

Medium

Issue Description

The total/global poolWeight is recorded in `poolWeight[address(0)]` .

It has already been updated each time `updateEpoch()` is called and the epoch is pushed forward.

Anyone can call `updateInfoAboutStakedNumber(address(0))` to inflate `poolWeight[address(0)]` as it is a public function and there is no validation on `collection` .

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L1292-L1307

```
1292  function updateInfoAboutStakedNumber(address collection) public returns (uint256
      actualWeight)
1293  {
1294      uint256 lastUpdateEpoch = lastUpdatesOfStakedNumbers[collection];
1295      if (lastUpdateEpoch == currentEpoch)
1296          return poolWeight[collection][currentEpoch];
1297
1298      uint256 i = lastUpdateEpoch + 1;
1299      for (; i <= currentEpoch; ++i)
1300      {
1301          numberOfStakedNftsInCollection[i][collection] +=
      numberOfStakedNftsInCollection[i - 1][collection];
1302          poolWeight[collection][i] += poolWeight[collection][i - 1];
1303      }
1304
1305      lastUpdatesOfStakedNumbers[collection] = currentEpoch;
1306      return poolWeight[collection][currentEpoch];
1307  }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L1182-L1202

```
1182  function updateEpoch() public {
1183      require(getCurrentStage() == Stage.FifthStage, "Wrong stage!");             //
      Requires to be at fourth stage.
1184      require(block.timestamp >= epochStartDate + epochDuration ||
      numberOfPlayedPairsInEpoch[currentEpoch] == pairsInEpoch[currentEpoch].length); //
      Requires fourth stage to end, or determine every pair winner.
1185
1186      zooFunctions = IZooFunctions(zooGovernance.zooFunctions());                 //
      Sets ZooFunctions to contract specified in zooGovernance.
1187
1188      epochStartDate = block.timestamp;                                           //
      Sets start date of new epoch.
1189      currentEpoch++;                                                             //
      Increments currentEpoch.
1190      epochsStarts[currentEpoch] = block.timestamp;                               //
      Records timestamp of new epoch start for ve-Zoo.
1191      nftsInGame = 0;                                                             //
      Nullifies amount of paired nfts.
1192      poolWeight[address(0)][currentEpoch] += poolWeight[address(0)][currentEpoch -
      1];
1193
1194      numberOfNftsWithNonZeroVotes += numberOfNftsWithNonZeroVotesPending;
1195      numberOfNftsWithNonZeroVotesPending = 0;
1196
1197      zooFunctions.resetRandom();     // Resets random in zoo functions.
1198
1199      (firstStageDuration, secondStageDuration, thirdStageDuration,
      fourthStageDuration, fifthStageDuration, epochDuration) =
      zooFunctions.getStageDurations();
1200
1201      emit EpochUpdated(block.timestamp, currentEpoch);
1202  }
```

Status

✓ Fixed

[WP-L7] Unnecessary negative `yTokensSaldo` in `_calculateBattleRewards()`

Low

Issue Description

L1113 is not necessary and wrong as the loser does not own any rewards.

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L1069-L1128

```
1069  function _calculateBattleRewards(uint256 winner, uint256 loser) internal
1070  {
1071      BattleRewardForEpoch storage winnerRewards =
      rewardsForEpoch[winner][currentEpoch];
1072      BattleRewardForEpoch storage loserRewards =
      rewardsForEpoch[loser][currentEpoch];
1073
1074      BattleRewardForEpoch storage winnerRewards1 =
      rewardsForEpoch[winner][currentEpoch + 1];
1075      BattleRewardForEpoch storage loserRewards1 =
      rewardsForEpoch[loser][currentEpoch + 1];
1076
1077      if (winner == 0 || loser == 0) // arena 50-50 case
1078      {
1079          if (winner == 0) { // Battle Arena won
1080              // Take yield
1081              loserRewards.isWinnerChose = true;
1082              uint256 income = loserRewards.yTokens -
      tokensToShares(loserRewards.tokensAtBattleStart);
1083              require(vault.redeem(income) == 0);
1084              _stablecoinTransfer(treasury, dai.balanceOf(address(this)));
1085          } else {
1086          // Grant Zoo
1087              winnerRewards.zooRewards +=
      zooFunctions.getLeagueZooRewards(winnerRewards.league);
1088              winnerRewards.isWinnerChose = true;
1089          }
1090          return;
1091      }
```

```
1092
1093        // Skip if price per share didn't change since pairing
1094        uint256 currentPps = vault.exchangeRateCurrent();
1095        if (winnerRewards.pricePerShareAtBattleStart == currentPps)
1096        {
1097            return;
1098        }
1099
1100        winnerRewards.pricePerShareCoef = currentPps *
    winnerRewards.pricePerShareAtBattleStart / (currentPps -
    winnerRewards.pricePerShareAtBattleStart);
1101        loserRewards.pricePerShareCoef = winnerRewards.pricePerShareCoef;
1102
1103        // Income = yTokens at battle end - yTokens at battle start
1104        uint256 income1 = winnerRewards.yTokens -
    tokensToShares(winnerRewards.tokensAtBattleStart);
1105        uint256 income2 = loserRewards.yTokens -
    tokensToShares(loserRewards.tokensAtBattleStart);
1106
1107        require(vault.redeem(((income1 + income2) / 25)) == 0);           // Withdraws
    dai from vault for yTokens, minus staker %.
1108
1109        uint256 daiReward = dai.balanceOf(address(this));
1110        _stablecoinTransfer(treasury, daiReward);
    // Transfers treasury part. 4 / 100 == 4%
1111
1112        winnerRewards.yTokensSaldo += int256(((income1 + income2) * 96 / 100));
1113        loserRewards.yTokensSaldo -= int256(income2);
1114
1115        winnerRewards1.yTokens = winnerRewards.yTokens + income2 - ((income1 +
    income2) / 25);
1116        loserRewards1.yTokens = loserRewards.yTokens - income2; // Withdraw reward
    amount.
1117
1118        stakingPositionsValues[winner].lastUpdateEpoch = currentEpoch + 1;          //
    Update lastUpdateEpoch to next epoch.
1119        stakingPositionsValues[loser].lastUpdateEpoch = currentEpoch + 1;           //
    Update lastUpdateEpoch to next epoch.
1120        winnerRewards1.votes += winnerRewards.votes;
    // Update votes for next epoch.
1121        loserRewards1.votes += loserRewards.votes;
    // Update votes for next epoch.
1122
```

```
1123        winnerRewards1.league = zooFunctions.getNftLeague(winnerRewards1.votes);    //
    Update league for next epoch.
1124        loserRewards1.league =
    zooFunctions.getNftLeague(loserRewards1.votes);        // Update league for next
    epoch.
1125
1126        winnerRewards.isWinnerChose = true;
1127        loserRewards.isWinnerChose = true;
1128    }
```

Status

✓ Fixed

[WP-L8] Attackers can grief other players by quickly calling `chooseWinnerInPair()` to settle all games at `FifthStage`, preventing them from calling `removeVotesFromVeZoo()`.

Low

Issue Description

Since `removeVotesFromVeZoo()` can only be called at `FifthStage`, this causes the inability to retrieve `VeZoo` if `FifthStage` is very short.

Given the cost and impact of such an attack, we consider this a low severity issue, and you may choose not to apply any fix.

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L650-L658

```
650  function removeVotesFromVeZoo(address collection, uint256 amount) external
     only(address(veZoo))
651  {
652      require(getCurrentStage() == Stage.FifthStage, "Wrong stage!");
653
654      updateInfoAboutStakedNumber(collection);
655      poolWeight[collection][currentEpoch] -= amount * zooVoteRateNominator /
     zooVoteRateDenominator;
656      poolWeight[address(0)][currentEpoch] -= amount * zooVoteRateNominator /
     zooVoteRateDenominator;
657  }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L1182-L1203

```
1182  function updateEpoch() public {
1183      require(getCurrentStage() == Stage.FifthStage, "Wrong stage!");           //
      Requires to be at fourth stage.
1184      require(block.timestamp >= epochStartDate + epochDuration ||
      numberOfPlayedPairsInEpoch[currentEpoch] == pairsInEpoch[currentEpoch].length); //
      Requires fourth stage to end, or determine every pair winner.
```

```
1185
1186        zooFunctions = IZooFunctions(zooGovernance.zooFunctions());                 //
    Sets ZooFunctions to contract specified in zooGovernance.
1187
1188        epochStartDate = block.timestamp;                                           //
    Sets start date of new epoch.
1189        currentEpoch++;                                                             //
    Increments currentEpoch.
1190        epochsStarts[currentEpoch] = block.timestamp;                               //
    Records timestamp of new epoch start for ve-Zoo.
1191        nftsInGame = 0;                                                             //
    Nullifies amount of paired nfts.
1192        poolWeight[address(0)][currentEpoch] += poolWeight[address(0)][currentEpoch -
    1];
1193
1194        numberOfNftsWithNonZeroVotes += numberOfNftsWithNonZeroVotesPending;
1195        numberOfNftsWithNonZeroVotesPending = 0;
1196
1197        zooFunctions.resetRandom();     // Resets random in zoo functions.
1198
1199        (firstStageDuration, secondStageDuration, thirdStageDuration,
    fourthStageDuration, fifthStageDuration, epochDuration) =
    zooFunctions.getStageDurations();
1200
1201        emit EpochUpdated(block.timestamp, currentEpoch);
1202    }
```

Status

✓ Fixed

[WP-H9] `votingPosition.lastEpochYTokensWereDeductedForRewards` is not handled correctly.

High

Issue Description

1. `addDaiToVoting()` won't update `votingPosition.lastEpochYTokensWereDeductedForRewards`, resulting in the repeated reduction of `votingPosition.yTokensNumber` when `withdrawDaiFromVoting() -> _subtractYTokensUserForRewardsFromVotingPosition()` is called.
2. `votingPosition.lastEpochYTokensWereDeductedForRewards` is not initialized, causing every votingPosition to start stripping interests/rewards from epoch 0 regardless of its start epoch.

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L512-L558

```
512     function addDaiToVoting(uint256 votingPositionId, address voter, uint256
    amount, uint256 _yTokens) public only(nftVotingPosition) returns (uint256 votes)
513     {
514         require(getCurrentStage() != Stage.ThirdStage, "Wrong stage!");
515 
516         VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
517         uint256 stakingPositionId = votingPosition.stakingPositionId;
    // Gets id of staker position.
518         require(stakingPositionsValues[stakingPositionId].endEpoch == 0, "E1");
    // Requires to be staked.
519 
520         _updateVotingPosition(votingPositionId);
521         // _updateVotingRewardDebt(votingPositionId);
522 
523         votes = zooFunctions.computeVotesByDai(amount);
    // Gets computed amount of votes from multiplier of dai.
524         // case for NOT swap.
525         if (_yTokens == 0)
    // if no _yTokens from another position with swap.
526         {
```

```
527             _yTokens = vault.balanceOf(address(this));
528             require(vault.mint(amount) == 0);
// Deposits dai to yearn and gets yTokens.
529             _yTokens = vault.balanceOf(address(this)) - _yTokens;
530         }
531
532         uint256 epoch = currentEpoch;
533         if (getCurrentStage() > Stage.SecondStage)
534         {
535             epoch += 1;
536             pendingVotes[votingPositionId] += votes;
537             pendingVotesEpoch[votingPositionId] = currentEpoch;
538         }
539         else
540         {
541             votingPosition.daiVotes += votes;
// Adds computed daiVotes amount from to voting position.
542             votingPosition.votes += votes;
// Adds computed votes amount to totalVotes amount for voting position.
543         }
544
545         votingPosition.yTokensNumber =
_calculateVotersYTokensExcludingRewards(votingPositionId) + _yTokens;// Adds
yTokens to voting position.
546         votingPosition.daiInvested += amount;
// Adds amount of dai to voting position.
547         votingPosition.startEpoch = epoch;
548
549         updateInfo(stakingPositionId);
550         BattleRewardForEpoch storage battleReward =
rewardsForEpoch[stakingPositionId][epoch];
551
552         battleReward.votes += votes;              // Adds votes to staker position
for current epoch.
553         battleReward.yTokens += _yTokens;         // Adds yTokens to rewards from
staker position for current epoch.
554
555         battleReward.league = zooFunctions.getNftLeague(battleReward.votes);
556
557         emit AddedDaiToVoting(currentEpoch, voter, stakingPositionId,
votingPositionId, amount, votes);
558     }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L724-L741

```
724     function _calculateVotersYTokensExcludingRewards(uint256 votingPositionId)
    internal view returns(uint256 yTokens)
725     {
726         VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
727         uint256 stakingPositionId = votingPosition.stakingPositionId;
728 
729         yTokens = votingPosition.yTokensNumber;
730         uint256 endEpoch = computeLastEpoch(votingPositionId);
731 
732         // From user yTokens subtract all tokens that go to the rewards
733         // This way allows to withdraw exact same amount of DAI user invested at
    the start
734         for (uint256 i = votingPosition.lastEpochYTokensWereDeductedForRewards; i
    < endEpoch; ++i)
735         {
736             if (rewardsForEpoch[stakingPositionId][i].pricePerShareCoef != 0)
737             {
738                 yTokens -= votingPosition.daiInvested * 10**18 /
    rewardsForEpoch[stakingPositionId][i].pricePerShareCoef;
739             }
740         }
741     }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L383-L436

```
383     function _createVotingPosition(uint256 stakingPositionId, address voter,
    uint256 yTokens, uint256 amount) public only(nftVotingPosition) returns (uint256
    votes, uint256 votingPositionId)
384     {
385         StakerPosition storage stakingPosition =
    stakingPositionsValues[stakingPositionId];
386         require(stakingPosition.startEpoch != 0 && stakingPosition.endEpoch == 0,
    "E1"); // Requires for staking position to be staked.
387 
388         VotingPosition storage position =
    votingPositionsValues[numberOfVotingPositions];
```

```
389         votes = zooFunctions.computeVotesByDai(amount);
    // Calculates amount of votes.
390
391         uint256 epoch = currentEpoch;
392         if (getCurrentStage() > Stage.ThirdStage)
393         {
394             epoch += 1;
395             pendingVotes[numberOfVotingPositions] = votes;
396             pendingVotesEpoch[numberOfVotingPositions] = currentEpoch;
397         }
398         else
399         {
400             position.daiVotes = votes;                      // Records computed
    amount of votes to daiVotes.
401             position.votes = votes;                         // Records computed
    amount of votes to total votes.
402         }
403
404         position.stakingPositionId = stakingPositionId;     // Records staker
    position Id voted for.
405         position.yTokensNumber = yTokens;                   // Records amount of
    yTokens got from yearn vault.
406         position.daiInvested = amount;                      // Records amount of
    dai invested.
407         position.startEpoch = epoch;                        // Records epoch when
    position created.
408         position.lastRewardedEpoch = epoch;                 // Sets starting point
    for reward to current epoch.
409         position.lastEpochOfIncentiveReward = epoch;        // Sets starting point
    for incentive rewards calculation.
410
411         BattleRewardForEpoch storage battleReward =
    rewardsForEpoch[stakingPositionId][currentEpoch];
412         BattleRewardForEpoch storage battleReward1 =
    rewardsForEpoch[stakingPositionId][epoch];
413
414         if (battleReward.votes == 0)
    // If staker position had zero votes before,
415         {
416             if (epoch == currentEpoch) // if vote for this epoch
417             {
418                 _swapActiveStakerPositions(stakingPositionId);
419                 numberOfNftsWithNonZeroVotes++;
```

```
420             }
421             else if (battleReward1.votes == 0) // if vote for next epoch and
    position have zero votes in both epochs.
422             {
423                 _swapActiveStakerPositions(stakingPositionId);
424                 numberOfNftsWithNonZeroVotesPending++;
425             }
426         }
427         battleReward1.votes += votes;
    // Adds votes for staker position for this epoch.
428         battleReward1.yTokens += yTokens;
    // Adds yTokens for this staker position for this epoch.
429
430         battleReward1.league = zooFunctions.getNftLeague(battleReward1.votes);
431
432         votingPositionId = numberOfVotingPositions;
433         numberOfVotingPositions++;
434
435         emit CreatedVotingPosition(epoch, voter, stakingPositionId, amount, votes,
    votingPositionId);
436     }
```

Recommendation

Given that `startEpoch` is never actually used, and `lastEpochYTokensWereDeductedForRewards` is a more appropriate name, consider changing to:

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L383-L436

```
383     function _createVotingPosition(uint256 stakingPositionId, address voter,
    uint256 yTokens, uint256 amount) public only(nftVotingPosition) returns (uint256
    votes, uint256 votingPositionId)
384     {
385         StakerPosition storage stakingPosition =
    stakingPositionsValues[stakingPositionId];
386         require(stakingPosition.startEpoch != 0 && stakingPosition.endEpoch == 0,
    "E1"); // Requires for staking position to be staked.
387
388         VotingPosition storage position =
    votingPositionsValues[numberOfVotingPositions];
```

```
389         votes = zooFunctions.computeVotesByDai(amount);
    // Calculates amount of votes.
390
391         uint256 epoch = currentEpoch;
392         if (getCurrentStage() > Stage.ThirdStage)
393         {
394             epoch += 1;
395             pendingVotes[numberOfVotingPositions] = votes;
396             pendingVotesEpoch[numberOfVotingPositions] = currentEpoch;
397         }
398         else
399         {
400             position.daiVotes = votes;                      // Records computed
    amount of votes to daiVotes.
401             position.votes = votes;                         // Records computed
    amount of votes to total votes.
402         }
403
404         position.stakingPositionId = stakingPositionId;     // Records staker
    position Id voted for.
405         position.yTokensNumber = yTokens;                   // Records amount of
    yTokens got from yearn vault.
406         position.daiInvested = amount;                      // Records amount of
    dai invested.
407         position.lastEpochYTokensWereDeductedForRewards = epoch;
    // Records epoch when position created.
408         position.lastRewardedEpoch = epoch;                 // Sets starting point
    for reward to current epoch.
409         position.lastEpochOfIncentiveReward = epoch;        // Sets starting point
    for incentive rewards calculation.
410
411         BattleRewardForEpoch storage battleReward =
    rewardsForEpoch[stakingPositionId][currentEpoch];
412         BattleRewardForEpoch storage battleReward1 =
    rewardsForEpoch[stakingPositionId][epoch];
413
414         if (battleReward.votes == 0)
    // If staker position had zero votes before,
415         {
416             if (epoch == currentEpoch) // if vote for this epoch
417             {
418                 _swapActiveStakerPositions(stakingPositionId);
419                 numberOfNftsWithNonZeroVotes++;
```

```
420             }
421             else if (battleReward1.votes == 0) // if vote for next epoch and
    position have zero votes in both epochs.
422             {
423                 _swapActiveStakerPositions(stakingPositionId);
424                 numberOfNftsWithNonZeroVotesPending++;
425             }
426         }
427         battleReward1.votes += votes;
    // Adds votes for staker position for this epoch.
428         battleReward1.yTokens += yTokens;
    // Adds yTokens for this staker position for this epoch.
429
430         battleReward1.league = zooFunctions.getNftLeague(battleReward1.votes);
431
432         votingPositionId = numberOfVotingPositions;
433         numberOfVotingPositions++;
434
435         emit CreatedVotingPosition(epoch, voter, stakingPositionId, amount, votes,
    votingPositionId);
436     }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L512-L558

```
512     function addDaiToVoting(uint256 votingPositionId, address voter, uint256
    amount, uint256 _yTokens) public only(nftVotingPosition) returns (uint256 votes)
513     {
514         require(getCurrentStage() != Stage.ThirdStage, "Wrong stage!");
515
516         VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
517         uint256 stakingPositionId = votingPosition.stakingPositionId;
    // Gets id of staker position.
518         require(stakingPositionsValues[stakingPositionId].endEpoch == 0, "E1");
    // Requires to be staked.
519
520         _updateVotingPosition(votingPositionId);
521         // _updateVotingRewardDebt(votingPositionId);
522
523         votes = zooFunctions.computeVotesByDai(amount);
    // Gets computed amount of votes from multiplier of dai.
```

```
524         // case for NOT swap.
525         if (_yTokens == 0)
    // if no _yTokens from another position with swap.
526         {
527             _yTokens = vault.balanceOf(address(this));
528             require(vault.mint(amount) == 0);
    // Deposits dai to yearn and gets yTokens.
529             _yTokens = vault.balanceOf(address(this)) - _yTokens;
530         }
531
532         uint256 epoch = currentEpoch;
533         if (getCurrentStage() > Stage.SecondStage)
534         {
535             epoch += 1;
536             pendingVotes[votingPositionId] += votes;
537             pendingVotesEpoch[votingPositionId] = currentEpoch;
538         }
539         else
540         {
541             votingPosition.daiVotes += votes;
    // Adds computed daiVotes amount from to voting position.
542             votingPosition.votes += votes;
    // Adds computed votes amount to totalVotes amount for voting position.
543         }
544
545         votingPosition.yTokensNumber =
    _calculateVotersYTokensExcludingRewards(votingPositionId) + _yTokens;// Adds
    yTokens to voting position.
546         votingPosition.daiInvested += amount;
    // Adds amount of dai to voting position.
547         votingPosition.lastEpochYTokensWereDeductedForRewards = epoch;
548
549         updateInfo(stakingPositionId);
550         BattleRewardForEpoch storage battleReward =
    rewardsForEpoch[stakingPositionId][epoch];
551
552         battleReward.votes += votes;              // Adds votes to staker position
    for current epoch.
553         battleReward.yTokens += _yTokens;         // Adds yTokens to rewards from
    staker position for current epoch.
554
555         battleReward.league = zooFunctions.getNftLeague(battleReward.votes);
556
```

```
557           emit AddedDaiToVoting(currentEpoch, voter, stakingPositionId,
      votingPositionId, amount, votes);
558       }
```

Status

✓ Fixed

[WP-H10] When `stakerReward` is not claimed, `saldo` is part of the principal for each round's calculation.

High

Issue Description

However, when `staker` calls `claimRewardFromStaking()` , the principal for the current round is not deducted as expected from `yTokens` (`rewardsForEpoch[stakingPositionId][epoch].yTokens`). This causes `yTokens` to be higher than it should be.

Similar deduction is done at L834 for voter rewards.

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L908-L922

```
908     function claimRewardFromStaking(uint256 stakingPositionId, address staker,
    address beneficiary) public only(nftStakingPosition) returns (uint256 daiReward)
909     {
910         StakerPosition storage stakerPosition =
    stakingPositionsValues[stakingPositionId];
911         require(getCurrentStage() == Stage.FirstStage || stakerPosition.endEpoch
    != 0, "Wrong stage!"); // Requires to be at first stage in battle epoch.
912 
913         updateInfo(stakingPositionId);
914         (uint256 yTokenReward, uint256 end) =
    getPendingStakerReward(stakingPositionId);
915         stakerPosition.lastRewardedEpoch = end;
    // Records epoch of last reward claim.
916 
917         require(vault.redeem(yTokenReward) == 0);
    // Gets reward from yearn.
918         daiReward = dai.balanceOf(address(this));
919         _stablecoinTransfer(beneficiary, daiReward);
920 
921         emit ClaimedRewardFromStaking(currentEpoch, staker, stakingPositionId,
    beneficiary, yTokenReward, daiReward);
922     }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L927-L943

```
927     function getPendingStakerReward(uint256 stakingPositionId) public view returns
    (uint256 stakerReward, uint256 end)
928     {
929         StakerPosition storage stakerPosition =
    stakingPositionsValues[stakingPositionId];
930         uint256 endEpoch = stakerPosition.endEpoch;
    // Gets endEpoch from position.
931 
932         end = endEpoch == 0 ? currentEpoch : endEpoch;
    // Sets end variable to endEpoch if it non-zero, otherwise to currentEpoch.
933 
934         for (uint256 i = stakerPosition.lastRewardedEpoch; i < end; ++i)
935         {
936             int256 saldo = rewardsForEpoch[stakingPositionId][i].yTokensSaldo;
    // Get saldo from staker position.
937 
938             if (saldo > 0)
939             {
940                 stakerReward += uint256(saldo / 96);
    // Calculates reward for staker: 1% = 1 / 96
941             }
942         }
943     }
```

Status

✓ Fixed

[WP-H11] Unexpected duplication of interest allocation on `saldo`, leading to inflated BattleRewards calculation

High

Issue Description

Under the current implementation, the income (number of shares) from participating in the game is entirely distributed as raward, including the interest generated by these shares. The treasury part is directly transferred (L1110), while the rest is recorded as PendingReward in `rewardsForEpoch[stakingPositionId][epoch].yTokensSaldo` (L1112) and is entirely taken by voters/stakers.

However, these income shares are also recorded in `rewardsForEpoch[stakingPositionId][epoch].yTokens` (L1115). `yTokens` includes PendingReward, and as the entire `yTokens` participates in the interest game of subsequent epochs, the interest distribution of PendingReward is duplicated with the distribution described in the previous paragraph.

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L1066-L1128

```
1066      /// @dev Contains calculation logic of battle rewards
1067      /// @param winner stakingPositionId of NFT that WON in battle
1068      /// @param loser stakingPositionId of NFT that LOST in battle
1069      function _calculateBattleRewards(uint256 winner, uint256 loser) internal
1070      {
1071          BattleRewardForEpoch storage winnerRewards =
      rewardsForEpoch[winner][currentEpoch];
1072          BattleRewardForEpoch storage loserRewards =
      rewardsForEpoch[loser][currentEpoch];
1073
1074          BattleRewardForEpoch storage winnerRewards1 =
      rewardsForEpoch[winner][currentEpoch + 1];
1075          BattleRewardForEpoch storage loserRewards1 =
      rewardsForEpoch[loser][currentEpoch + 1];
1076
1077          if (winner == 0 || loser == 0) // arena 50-50 case
1078          {
```

```
1079            if (winner == 0) { // Battle Arena won
1080                // Take yield
1081                loserRewards.isWinnerChose = true;
1082                uint256 income = loserRewards.yTokens -
    tokensToShares(loserRewards.tokensAtBattleStart);
1083                require(vault.redeem(income) == 0);
1084                _stablecoinTransfer(treasury, dai.balanceOf(address(this)));
1085            } else {
1086            // Grant Zoo
1087                winnerRewards.zooRewards +=
    zooFunctions.getLeagueZooRewards(winnerRewards.league);
1088                winnerRewards.isWinnerChose = true;
1089            }
1090            return;
1091        }
1092
1093        // Skip if price per share didn't change since pairing
1094        uint256 currentPps = vault.exchangeRateCurrent();
1095        if (winnerRewards.pricePerShareAtBattleStart == currentPps)
1096        {
1097            return;
1098        }
1099
1100        winnerRewards.pricePerShareCoef = currentPps *
    winnerRewards.pricePerShareAtBattleStart / (currentPps -
    winnerRewards.pricePerShareAtBattleStart);
1101        loserRewards.pricePerShareCoef = winnerRewards.pricePerShareCoef;
1102
1103        // Income = yTokens at battle end - yTokens at battle start
1104        uint256 income1 = winnerRewards.yTokens -
    tokensToShares(winnerRewards.tokensAtBattleStart);
1105        uint256 income2 = loserRewards.yTokens -
    tokensToShares(loserRewards.tokensAtBattleStart);
1106
1107        require(vault.redeem(((income1 + income2) / 25)) == 0);           //
    Withdraws dai from vault for yTokens, minus staker %.
1108
1109        uint256 daiReward = dai.balanceOf(address(this));
1110        _stablecoinTransfer(treasury, daiReward);
    // Transfers treasury part. 4 / 100 == 4%
1111
1112        winnerRewards.yTokensSaldo += int256(((income1 + income2) * 96 / 100));
1113        loserRewards.yTokensSaldo -= int256(income2);
```

```
1114
1115        winnerRewards1.yTokens = winnerRewards.yTokens + income2 - ((income1 +
    income2) / 25);
1116        loserRewards1.yTokens = loserRewards.yTokens - income2; // Withdraw reward
    amount.
1117
1118        stakingPositionsValues[winner].lastUpdateEpoch = currentEpoch + 1;
    // Update lastUpdateEpoch to next epoch.
1119        stakingPositionsValues[loser].lastUpdateEpoch = currentEpoch + 1;
    // Update lastUpdateEpoch to next epoch.
1120        winnerRewards1.votes += winnerRewards.votes;
    // Update votes for next epoch.
1121        loserRewards1.votes += loserRewards.votes;
    // Update votes for next epoch.
1122
1123        winnerRewards1.league =
    zooFunctions.getNftLeague(winnerRewards1.votes);    // Update league for next
    epoch.
1124        loserRewards1.league =
    zooFunctions.getNftLeague(loserRewards1.votes);        // Update league for next
    epoch.
1125
1126        winnerRewards.isWinnerChose = true;
1127        loserRewards.isWinnerChose = true;
1128    }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L806-L846

```
806     /// @notice Function to claim reward in yTokens from voting.
807     /// @param votingPositionId - id of voting position.
808     /// @param beneficiary - address of recipient of reward.
809     function claimRewardFromVoting(uint256 votingPositionId, address voter,
    address beneficiary) external only(nftVotingPosition) returns (uint256 daiReward)
810     {
811         VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
812
813         require(getCurrentStage() == Stage.FirstStage ||
    stakingPositionsValues[votingPosition.stakingPositionId].endEpoch != 0, "Wrong
    stage!"); // Requires to be at first stage or position should be liquidated.
```

```
814
815         updateInfo(votingPosition.stakingPositionId);
816
817         (uint256 yTokenReward, uint256 zooRewards) =
    getPendingVoterReward(votingPositionId); // Calculates amount of reward in
    yTokens.
818
819         yTokenReward += votingPosition.yTokensRewardDebt;
    // Adds reward debt, from previous epochs.
820         zooRewards += zooTokensRewardDebt[votingPositionId];
821         votingPosition.yTokensRewardDebt = 0;
    // Nullify reward debt.
822         zooTokensRewardDebt[votingPositionId] = 0;
823
824         yTokenReward = yTokenReward * 95 / 96; // 95% of income to voter.
825
826         require(vault.redeem(yTokenReward) == 0);
    // Withdraws dai from vault for yTokens, minus staker %.
827         daiReward = dai.balanceOf(address(this));
828
829         _stablecoinTransfer(beneficiary, daiReward);
    // Transfers voter part of reward.
830
831         BattleRewardForEpoch storage battleReward =
    rewardsForEpoch[votingPosition.stakingPositionId][currentEpoch];
832         if (battleReward.yTokens >= yTokenReward)
833         {
834             battleReward.yTokens -= yTokenReward;
    // Subtracts yTokens for this position.
835         }
836         else
837         {
838             battleReward.yTokens = 0;
839         }
840
841         zoo.transfer(beneficiary, zooRewards);
842
843         votingPosition.lastRewardedEpoch = computeLastEpoch(votingPositionId);
    // Records epoch of last reward claimed.
844
845         emit ClaimedRewardFromVoting(currentEpoch, voter,
    votingPosition.stakingPositionId, beneficiary, daiReward, votingPositionId);
846     }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L866-L903

```
866     /// @notice Function to calculate pending reward from voting for position with this id.
867     /// @param votingPositionId - id of voter position in battles.
868     /// @return yTokens - amount of pending reward and 2 technical numbers, which must me always equal 0.
869     function getPendingVoterReward(uint256 votingPositionId) public view returns (uint256 yTokens, uint256 zooRewards)
870     {
871         VotingPosition storage votingPosition = votingPositionsValues[votingPositionId];
872 
873         uint256 endEpoch = computeLastEpoch(votingPositionId);
874 
875         uint256 stakingPositionId = votingPosition.stakingPositionId; // Gets staker position id from voter position.
876 
877         uint256 pendingVotes = pendingVotes[votingPositionId];
878         uint256 pendingVotesEpoch = pendingVotesEpoch[votingPositionId];
879         uint256 votes = votingPosition.votes;
880         for (uint256 i = votingPosition.lastRewardedEpoch; i < endEpoch; ++i)
881         {
882             if (i == pendingVotesEpoch + 1 && pendingVotes > 0)
883             {
884                 votes += pendingVotes;
885             }
886 
887             int256 saldo = rewardsForEpoch[stakingPositionId][i].yTokensSaldo; // Gets saldo from staker position for every epoch in range.
888 
889             if (saldo > 0)
890             {
891                 yTokens += uint256(saldo) * votes / rewardsForEpoch[stakingPositionId][i].votes;        // Calculates yTokens amount for voter.
892             }
893 
894             BattleRewardForEpoch storage leagueRewards = rewardsForEpoch[stakingPositionId][i];
```

```
895
896             if (rewardsForEpoch[stakingPositionId][i].votes > 0)
897             {
898                 zooRewards += leagueRewards.zooRewards * votes /
    rewardsForEpoch[stakingPositionId][i].votes;         // Calculates yTokens amount
    for voter.
899             }
900         }
901
902         return (yTokens, zooRewards);
903     }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L945-L1021

```
945     /// @notice Function for pair nft for battles.
946     /// @param stakingPositionId - id of staker position.
947     function pairNft(uint256 stakingPositionId) external
948     {
949         require(getCurrentStage() == Stage.ThirdStage, "Wrong stage!");
    // Requires to be at 3 stage of battle epoch.
950
951         updateInfo(stakingPositionId);
952         BattleRewardForEpoch storage battleReward1 =
    rewardsForEpoch[stakingPositionId][currentEpoch];
953
954         // this require makes impossible to pair if there are no available pair.
    // require(numberOfNftsWithNonZeroVotes / 2 > nftsInGame / 2, "E1");            //
    Requires enough nft for pairing.
955         uint256 index1;
    // Index of nft paired for.
956         uint256[] memory leagueList = new uint256[](numberOfNftsWithNonZeroVotes);
957         uint256 nftsInSameLeague = 0;
958         bool idFound;
959
960         // Find first staking position and get list of opponents from league for
    index2
961         for (uint256 i = nftsInGame; i < numberOfNftsWithNonZeroVotes; ++i)
962         {
963             updateInfo(activeStakerPositions[i]);
964             if (activeStakerPositions[i] == stakingPositionId)
```

```
965             {
966                 index1 = i;
967                 idFound = true;
968                 continue;
969                 // break;
970             }
971             // In the same league
972             else if (battleReward1.league ==
    rewardsForEpoch[activeStakerPositions[i]][currentEpoch].league)
973             {
974                 leagueList[nftsInSameLeague] = activeStakerPositions[i];
975                 nftsInSameLeague++;
976             }
977         }
978         require(idFound, "E1");
979 
980         (activeStakerPositions[index1], activeStakerPositions[nftsInGame]) =
    (activeStakerPositions[nftsInGame], activeStakerPositions[index1]);// Swaps
    nftsInGame with index.
981         nftsInGame++;
    // Increases amount of paired nft.
982 
983         uint256 stakingPosition2;
984         battleReward1.tokensAtBattleStart = sharesToTokens(battleReward1.yTokens);
    // Records amount of yTokens on the moment of pairing for candidate.
985         battleReward1.pricePerShareAtBattleStart = vault.exchangeRateCurrent();
986 
987         if (nftsInSameLeague != 0)
988         {
989             uint256 index2;
990             stakingPosition2 = leagueList[0];
991             if (nftsInSameLeague > 1)
992             {
993                 stakingPosition2 = leagueList[zooFunctions.computePseudoRandom() %
    nftsInSameLeague];
994             }
995 
996             for (uint256 i = nftsInGame; i < numberOfNftsWithNonZeroVotes; ++i)
997             {
998                 if (activeStakerPositions[i] == stakingPosition2)
999                 {
1000                     index2 = i;
1001                 }
```

```
1002            }
1003
1004            //updateInfo(stakingPosition2);
1005            BattleRewardForEpoch storage battleReward2 =
    rewardsForEpoch[stakingPosition2][currentEpoch];
1006            battleReward2.tokensAtBattleStart =
    sharesToTokens(battleReward2.yTokens);           // Records amount of yTokens on
    the moment of pairing for opponent.
1007            battleReward2.pricePerShareAtBattleStart =
    vault.exchangeRateCurrent();
1008
1009            (activeStakerPositions[index2], activeStakerPositions[nftsInGame]) =
    (activeStakerPositions[nftsInGame], activeStakerPositions[index2]); // Swaps
    nftsInGame with index of opponent.
1010            nftsInGame++;
    // Increases amount of paired nft.
1011        }
1012        else
1013        {
1014            stakingPosition2 = 0;
1015        }
1016
1017        pairsInEpoch[currentEpoch].push(NftPair(stakingPositionId,
    stakingPosition2, false, false));// Pushes nft pair to array of pairs.
1018        uint256 pairIndex = getNftPairLength(currentEpoch) - 1;
1019
1020        emit PairedNft(currentEpoch, stakingPositionId, stakingPosition2,
    pairIndex);
1021    }
```

Recommendation

Ensure that `rewardsForEpoch[stakingPositionId][epoch].yTokens` ***does not include*** any leading/pending rewards, and that `rewardsForEpoch[stakingPositionId][epoch].yTokens` does not intersect with `rewardsForEpoch[stakingPositionId][epoch].yTokensSaldo` .

Status

✓ Fixed

[WP-H12] The miscalculation in `_calculateVotersYTokensExcludingRewards()` can lead to an inflated `votingPositionsValues[votingPositionId].yTokensNumber` .

High

Issue Description

The principal of the voter is expected not to increase in the formula, while the code actually allows the principal to increase.

L738 `yTokens -= votingPosition.daiInvested * 10**18 / rewardsForEpoch[stakingPositionId][i].pricePerShareCoef;`

ie.

based on:

```
Current round principal / Old pps of the current round = Starting shares of the current round
Starting shares of the current round * New pps - Starting shares of the current round * Old pps = Interest amount of the current round
Interest amount of the current round / New pps of the current round = Shares corresponding to the interest of the current round
```

`yTokens -= votingPosition.daiInvested * 10**18 / rewardsForEpoch[stakingPositionId][i].pricePerShareCoef;` can meet the implicit requirement of that the principal of each round remains unchanged as `votingPosition.daiInvested` .

However, `votingPosition.daiInvested` may not be the same in every round. This is because the interests accumulated during stage 1 and 2, as well as the epochs that the stakerPosition skipped (`rewardsForEpoch[stakingPositionId][i].pricePerShareCoe == 0`), did not participate in the games. Therefore, the starting principal of each epoch may not be `votingPosition.daiInvested` .

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L721-L741

```
721     /// @dev Calculates voting position's own yTokens - excludes yTokens that was used for rewards
722     /// @dev yTokens must be substracted even if voting won in battle (they go to the voting's pending reward)
723     /// @param votingPositionId ID of voting to calculate yTokens
724     function _calculateVotersYTokensExcludingRewards(uint256 votingPositionId) internal view returns(uint256 yTokens)
725     {
726         VotingPosition storage votingPosition = votingPositionsValues[votingPositionId];
727         uint256 stakingPositionId = votingPosition.stakingPositionId;
728 
729         yTokens = votingPosition.yTokensNumber;
730         uint256 endEpoch = computeLastEpoch(votingPositionId);
731 
732         // From user yTokens subtract all tokens that go to the rewards
733         // This way allows to withdraw exact same amount of DAI user invested at the start
734         for (uint256 i = votingPosition.lastEpochYTokensWereDeductedForRewards; i < endEpoch; ++i)
735         {
736             if (rewardsForEpoch[stakingPositionId][i].pricePerShareCoef != 0)
737             {
738                 yTokens -= votingPosition.daiInvested * 10**18 / rewardsForEpoch[stakingPositionId][i].pricePerShareCoef;
739             }
740         }
741     }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L659-L711

```
659     /// @dev Function to liquidate voting position and claim reward.
660     /// @param votingPositionId - id of position.
661     /// @param voter - address of position owner.
662     /// @param beneficiary - address of recipient.
663     /// @param stakingPositionId - id of staking position.
664     /// @param toSwap - boolean for swap votes, True if called from swapVotes function.
```

```
665     function _liquidateVotingPosition(uint256 votingPositionId, address voter,
    address beneficiary, uint256 stakingPositionId, bool toSwap) internal
666     {
667         VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
668 
669         uint256 yTokens = votingPosition.yTokensNumber;
670 
671         if (toSwap == false)                                        // If false,
    withdraws tokens from vault for regular liquidate.
672         {
673             require(vault.redeem(yTokens) == 0);
674             _stablecoinTransfer(beneficiary, dai.balanceOf(address(this))); //
    True when called from swapVotes, ignores withdrawal to re-assign them for another
    position.
675         }
676 
677         _withdrawZoo(votingPosition.zooInvested, beneficiary);
    // Even if it is swap, withdraws all zoo.
678 
679         votingPosition.endEpoch = currentEpoch;                     // Sets
    endEpoch to currentEpoch.
680 
681         BattleRewardForEpoch storage battleReward =
    rewardsForEpoch[stakingPositionId][currentEpoch];
682         battleReward.votes -= votingPosition.votes;                 // Decreases
    votes for staking position in current epoch.
683 
684         if (battleReward.yTokens >= yTokens)                        // If
    withdraws less than in staking position.
685         {
686             battleReward.yTokens -= yTokens;                        // Decreases
    yTokens for this staking position.
687         }
688         else
689         {
690             battleReward.yTokens = 0;                               // Or nullify
    it if trying to withdraw more yTokens than left in position(because of yTokens
    current rate)
691         }
692 
693         // IF there is votes on position AND staking position is active
694         if (battleReward.votes == 0 &&
    stakingPositionsValues[stakingPositionId].endEpoch == 0)
```

```
695         {
696             // Move staking position to part, where staked without votes.
697             for(uint256 i = 0; i < activeStakerPositions.length; ++i)
698             {
699                 if (activeStakerPositions[i] == stakingPositionId)
700                 {
701                     (activeStakerPositions[i],
    activeStakerPositions[numberOfNftsWithNonZeroVotes - 1]) =
    (activeStakerPositions[numberOfNftsWithNonZeroVotes - 1],
    activeStakerPositions[i]);      // Swaps position to end of array
702                     numberOfNftsWithNonZeroVotes--;
    // Decrements amount of non-zero positions.
703                     break;
704                 }
705             }
706         }
707
708         battleReward.league = zooFunctions.getNftLeague(battleReward.votes);
709
710         emit LiquidatedVotingPosition(currentEpoch, voter, stakingPositionId,
    beneficiary, votingPositionId, votingPosition.zooInvested * 995 / 1000,
    votingPosition.daiInvested);
711     }
```

Recommendation

Consider replacing `votingPosition.daiInvested` with `yTokens * pricePerShareAtBattleStart` .

Status

✓ Fixed

[WP-H13] Not properly maintaining `votingPositionsValues[votingPositionId].yTokensNumber` resulted in voters receiving less principal during `_liquidateVotingPosition()` (the remaining will be frozen in the NftBattleArena contract).

High

Issue Description

Expected:

During the FourthStage and FifthStage `addDaiToVoting()` scenarios, the `_yTokens` at L545 did not participate in the `currentEpoch` game, so it should not be reduced during the settlement of `currentEpoch` (in other words, the `rewardsForEpoch[stakingPositionId][currentEpoch].pricePerShareCoef` of the `currentEpoch` round should not be applied to this `_yTokens` portion).

Current implementation: Line 545 combines the `_yTokens` portion with the previously participated `yTokensNumber` in the `currentEpoch` game, resulting in an unintended reduction of this `_yTokens` portion during the settlement of `currentEpoch` .

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L507-L558

```
507     /// @notice Function to add dai tokens to voting position.
508     /// @param votingPositionId - id of voting position.
509     /// @param voter - address of voter.
510     /// @param amount - amount of dai tokens to add.
511     /// @param _yTokens - amount of yTokens from previous position when called
    with swap.
512     function addDaiToVoting(uint256 votingPositionId, address voter, uint256
    amount, uint256 _yTokens) public only(nftVotingPosition) returns (uint256 votes)
513     {
514         require(getCurrentStage() != Stage.ThirdStage, "Wrong stage!");
515
516         VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
```

```
517         uint256 stakingPositionId = votingPosition.stakingPositionId;
    // Gets id of staker position.
518         require(stakingPositionsValues[stakingPositionId].endEpoch == 0, "E1");
    // Requires to be staked.
519
520         _updateVotingPosition(votingPositionId);
521         // _updateVotingRewardDebt(votingPositionId);
522
523         votes = zooFunctions.computeVotesByDai(amount);
    // Gets computed amount of votes from multiplier of dai.
524         // case for NOT swap.
525         if (_yTokens == 0)
    // if no _yTokens from another position with swap.
526         {
527             _yTokens = vault.balanceOf(address(this));
528             require(vault.mint(amount) == 0);
    // Deposits dai to yearn and gets yTokens.
529             _yTokens = vault.balanceOf(address(this)) - _yTokens;
530         }
531
532         uint256 epoch = currentEpoch;
533         if (getCurrentStage() > Stage.SecondStage)
534         {
535             epoch += 1;
536             pendingVotes[votingPositionId] += votes;
537             pendingVotesEpoch[votingPositionId] = currentEpoch;
538         }
539         else
540         {
541             votingPosition.daiVotes += votes;
    // Adds computed daiVotes amount from to voting position.
542             votingPosition.votes += votes;
    // Adds computed votes amount to totalVotes amount for voting position.
543         }
544
545         votingPosition.yTokensNumber =
    _calculateVotersYTokensExcludingRewards(votingPositionId) + _yTokens;// Adds
    yTokens to voting position.
546         votingPosition.daiInvested += amount;
    // Adds amount of dai to voting position.
547         votingPosition.startEpoch = epoch;
548
549         updateInfo(stakingPositionId);
```

```
550         BattleRewardForEpoch storage battleReward =
    rewardsForEpoch[stakingPositionId][epoch];
551
552         battleReward.votes += votes;              // Adds votes to staker position
    for current epoch.
553         battleReward.yTokens += _yTokens;         // Adds yTokens to rewards from
    staker position for current epoch.
554
555         battleReward.league = zooFunctions.getNftLeague(battleReward.votes);
556
557         emit AddedDaiToVoting(currentEpoch, voter, stakingPositionId,
    votingPositionId, amount, votes);
558     }
```

https://github.com/ZooDAO-Project/arbitrum-battles/blob/0e12481210351665e1e5dc531a2e5a9ac1c63c34/contracts/NftBattleArena.sol#L659-L711

```
659     /// @dev Function to liquidate voting position and claim reward.
660     /// @param votingPositionId - id of position.
661     /// @param voter - address of position owner.
662     /// @param beneficiary - address of recipient.
663     /// @param stakingPositionId - id of staking position.
664     /// @param toSwap - boolean for swap votes, True if called from swapVotes
    function.
665     function _liquidateVotingPosition(uint256 votingPositionId, address voter,
    address beneficiary, uint256 stakingPositionId, bool toSwap) internal
666     {
667         VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
668
669         uint256 yTokens = votingPosition.yTokensNumber;
670
671         if (toSwap == false)                                    // If false,
    withdraws tokens from vault for regular liquidate.
672         {
673             require(vault.redeem(yTokens) == 0);
674             _stablecoinTransfer(beneficiary, dai.balanceOf(address(this))); //
    True when called from swapVotes, ignores withdrawal to re-assign them for another
    position.
675         }
676
```

```
677         _withdrawZoo(votingPosition.zooInvested, beneficiary);
    // Even if it is swap, withdraws all zoo.
678
679         votingPosition.endEpoch = currentEpoch;                    // Sets
    endEpoch to currentEpoch.
680
681         BattleRewardForEpoch storage battleReward =
    rewardsForEpoch[stakingPositionId][currentEpoch];
682         battleReward.votes -= votingPosition.votes;                // Decreases
    votes for staking position in current epoch.
683
684         if (battleReward.yTokens >= yTokens)                       // If
    withdraws less than in staking position.
685         {
686             battleReward.yTokens -= yTokens;                       // Decreases
    yTokens for this staking position.
687         }
688         else
689         {
690             battleReward.yTokens = 0;                              // Or nullify
    it if trying to withdraw more yTokens than left in position(because of yTokens
    current rate)
691         }
692
693         // IF there is votes on position AND staking position is active
694         if (battleReward.votes == 0 &&
    stakingPositionsValues[stakingPositionId].endEpoch == 0)
695         {
696             // Move staking position to part, where staked without votes.
697             for(uint256 i = 0; i < activeStakerPositions.length; ++i)
698             {
699                 if (activeStakerPositions[i] == stakingPositionId)
700                 {
701                     (activeStakerPositions[i],
    activeStakerPositions[numberOfNftsWithNonZeroVotes - 1]) =
    (activeStakerPositions[numberOfNftsWithNonZeroVotes - 1],
    activeStakerPositions[i]);      // Swaps position to end of array
702                     numberOfNftsWithNonZeroVotes--;
    // Decrements amount of non-zero positions.
703                     break;
704                 }
705             }
706         }
```

```
707
708         battleReward.league = zooFunctions.getNftLeague(battleReward.votes);
709
710         emit LiquidatedVotingPosition(currentEpoch, voter, stakingPositionId,
    beneficiary, votingPositionId, votingPosition.zooInvested * 995 / 1000,
    votingPosition.daiInvested);
711     }
```

PoC

1. Alice called `createNewVotingPosition()` and opened a position with $1 at Epoch 1, Stage 1.
2. Alice called `addDaiToVoting()` and added $100k to the position at Epoch 2, Stage 4.

While the newly added $100k should not participate in Epoch 2, in the current implementation, the **interest delta of Epoch 2** for the 100k will be unexpectedly stripped off.

Let's say the APR per epoch is 1%, the 100k which didn't participate in Epoch 2 should be deducted by 1%.

If Alice withdraws at Epoch 3, Stage 1, she should be able to withdraw slightly higher than 100k (plus the interest during Epoch 2, Stage 4 to Epoch 3, Stage 1).

In our current implementation, Alice can only withdraw a smaller amount than 100k because of the unexpected deduction of Epoch 2's interest.

Status

✓ **Fixed**

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.