ZooDAO Moonbeam Battle Audit Report

Feb 27, 2023

WATCHPUG

Table of Contents

Summary

This report has been prepared for ZooDAO Moonbeam Battle Audit Report 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 Moonbeam Battle Audit Report**
Codebase	**https://github.com/ZooDAO-Project/moonbeam-battles**
Commit	**c5cbb8eac31d9b39a5b11ceb109dd168c4566079**
Language	**Solidity**

Audit Summary

Delivery Date	**Feb 27, 2023**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**13**

[WP-H1] GLMR rewards are not being claimed

High

Issue Description

The GLMR rewards constitute 60% of the total APR (7.41% out of 11.48%) at the time of writing.

The current implemenation only claims the WELL rewards but not the GLMR rewards:

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L966-L975

```
function requestRandom() public
{
    require(getCurrentStage() == Stage.FifthStage, "Wrong stage!"); // Requires to be at 5th stage.

    uint256 wellInitialBalance = well.balanceOf(address(this));
    tokenController.claimReward(0, address(this));
    wellClaimedByEpoch[currentEpoch] = well.balanceOf(address(this)) - wellInitialBalance;

    zooFunctions.requestRandomNumber(); // Calls generate random number from chainlink or blockhash.
}
```

This is a big loss for all the users.

See:

https://github.com/moonwell-open-source/moonwell-contracts/blob/main/contracts/Comptroller.sol#L1253

Recommendation

GLMR rewards should also be claimed.

Status

✓ Fixed

[WP-H2] `NftBattleArena.updateInfoAboutStakedNumber()` Wrong calculation of `lastUpdatesOfStakedNumbers`

High

Issue Description

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L1188-L1197

```
function updateInfoAboutStakedNumber(address collection) public
{
    uint256 start = lastUpdatesOfStakedNumbers[collection] > 1 ? lastUpdatesOfStakedNumbers[collection] : 1;
    for (uint256 i = start; i <= currentEpoch; i++)
    {
        numberOfStakedNftsInCollection[i][collection] += numberOfStakedNftsInCollection[i - 1][collection];
    }

    lastUpdatesOfStakedNumbers[collection] = currentEpoch;
}
```

```
mapping (address => uint256) public lastUpdatesOfStakedNumbers;
```

`lastUpdatesOfStakedNumbers` records the `epoch` of the last update.

Therefore, when `lastUpdateEpoch == currentEpoch`, it should not go into the for loop. Otherwise, every time `updateInfoAboutStakedNumber()` is called, the code block inside the for loop will be executed for one extra time.

`updateInfoAboutStakedNumber()` can be called many times in the same epoch and `numberOfStakedNftsInCollection[epoch][collection] += numberOfStakedNftsInCollection[epoch - 1][collec` will be executed each time.

PoC

Given:

`removeStakerPosition()` is already called once in Epoch 2:

- `currentEpoch` : 2
- `lastUpdatesOfStakedNumbers[collection]` : 2
- `numberOfStakedNftsInCollection[1][collection]` : 10
- `numberOfStakedNftsInCollection[2][collection]` : 9

When: `removeStakerPosition()`

Then:

- L363 calls `updateInfoAboutStakedNumber(collection)`
- L1190 start = 2
- L1191 will go into the for loop as i = 2:
- L1193 `numberOfStakedNftsInCollection[2][collection] += numberOfStakedNftsInCollection[2 - 1][collectio`, ie, `numberOfStakedNftsInCollection[2][collection] += 10` . `numberOfStakedNftsInCollection[2][collection]` is updated from 9 to 19
- back to `removeStakerPosition()` L364 `numberOfStakedNftsInCollection[2][collection]` is updated from 19 to 18

Expected:

`numberOfStakedNftsInCollection[2][collection]` to be updated from `9` to `8` ;

Beacuse the previous `updateInfoAboutStakedNumber()` already updated to `currentEpoch` , it should not add the old value to `numberOfStakedNftsInCollection[2][collection]` again.

Recommendation

```
1188  function updateInfoAboutStakedNumber(address collection) public
1189  {
1190      uint256 lastUpdateEpoch = lastUpdatesOfStakedNumbers[collection]
1191      if (lastUpdateEpoch == currentEpoch)
1192          return;
```

```
    uint256 start = lastUpdateEpoch > 1 ? lastUpdateEpoch : 1;
    for (uint256 i = start; i <= currentEpoch; i++)
    {
        numberOfStakedNftsInCollection[i][collection] +=
numberOfStakedNftsInCollection[i - 1][collection];
    }

    lastUpdatesOfStakedNumbers[collection] = currentEpoch;
}
```

Status

✓ **Fixed**

[WP-H3] `updateInfo()` will override/discard the changes made in `_calculateBattleRewards()` because `lastUpdateEpoch` has not been changed at the same time

High

Issue Description

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L1048-L1057

```
function updateInfo(uint256 stakingPositionId) public
{
    uint256 lastUpdateEpoch = stakingPositionsValues[stakingPositionId].lastUpdateEpoch;
    if (lastUpdateEpoch == currentEpoch)
        return;

    rewardsForEpoch[stakingPositionId][currentEpoch].votes = rewardsForEpoch[stakingPositionId][lastUpdateEpoch].votes;
    rewardsForEpoch[stakingPositionId][currentEpoch].yTokens = rewardsForEpoch[stakingPositionId][lastUpdateEpoch].yTokens;
    stakingPositionsValues[stakingPositionId].lastUpdateEpoch = currentEpoch;
}
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L1011-L1044

```
function _calculateBattleRewards(uint256 winner, uint256 loser) internal
{
    BattleRewardForEpoch storage winnerRewards = rewardsForEpoch[winner][currentEpoch];
    BattleRewardForEpoch storage loserRewards = rewardsForEpoch[loser][currentEpoch];

    uint256 pps1 = winnerRewards.pricePerShareAtBattleStart;

    // Skip if price per share didn't change since pairing
```

```
1019        if (pps1 == vault.exchangeRateStored())
1020        {
1021            return;
1022        }
1023
1024        winnerRewards.pricePerShareCoef = vault.exchangeRateStored() * pps1 / (vault.exchangeRateStored() - pps1);
1025        loserRewards.pricePerShareCoef = winnerRewards.pricePerShareCoef;
1026
1027        // Income = yTokens at battle end - yTokens at battle start
1028        uint256 income1 = winnerRewards.yTokens - tokensToShares(winnerRewards.tokensAtBattleStart);
1029        uint256 income2 = loserRewards.yTokens - tokensToShares(loserRewards.tokensAtBattleStart);
1030
1031        uint256 xRewards = (income1 + income2) * 5 / 1000;
1032        uint256 jackpotRewards = (income1 + income2) * 1 / 100;
1033        vault.transfer(xZoo, xRewards); // todo: need to check that all is correct after that
1034        vault.transfer(jackpotA, jackpotRewards);
1035        vault.transfer(jackpotB, jackpotRewards);
1036        xZooRewards[currentEpoch] += xRewards;
1037        jackpotRewardsAtEpoch[currentEpoch] += jackpotRewards;
1038        winnerRewards.yTokensSaldo += int256(income1 + income2 - xRewards - 2 * jackpotRewards);
1039        loserRewards.yTokensSaldo -= int256(income2);
1040
1041
1042        rewardsForEpoch[winner][currentEpoch + 1].yTokens = winnerRewards.yTokens + income1 + income2 - xRewards - 2 * jackpotRewards;
1043        rewardsForEpoch[loser][currentEpoch + 1].yTokens = loserRewards.yTokens - income2;
1044    }
```

`chooseWinnerInPair() -> _calculateBattleRewards()` is supposed to settle the winning and losing for a pair, it updates `rewardsForEpoch[winner][currentEpoch + 1].yTokens` and `rewardsForEpoch[loser][currentEpoch + 1].yTokens` at L1042-1043.

However, it does not change their `lastUpdateEpoch` at the same time.

As a result, when `updateInfo()` is called for these accounts in the next epoch, their `yTokens`

will be reverted to the previous value: `rewardsForEpoch[stakingPositionId][lastUpdateEpoch].yTokens` instead of the updated value at L1042-1043.

Status

✓ **Fixed**

[WP-H4] Wrong interface for `mFRAX.redeemUnderlying()`

High

Issue Description

https://moonscan.io/address/0x1C55649f73CDA2f72CEf3DD6C5CA3d49EFcF484C#writeProxyContract

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/interfaces/IVault.sol#L9

```
function redeemUnderlying(uint256 redeemAmount, address recipient) external returns (uint256);
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L883-L894

```
function claimRewardFromStaking(uint256 stakingPositionId, address staker, address beneficiary) public only(nftStakingPosition) returns (uint256 daiReward)
{
    require(getCurrentStage() == Stage.FirstStage || stakingPositionsValues[stakingPositionId].endDate != 0, "Wrong stage!"); // Requires to be at first stage in battle epoch.

    updateInfo(stakingPositionId);
    (uint256 yTokenReward, uint256 end) = getPendingStakerReward(stakingPositionId);
    stakingPositionsValues[stakingPositionId].lastRewardedEpoch = end; // Records epoch of last reward claim.

    daiReward = vault.redeemUnderlying(yTokenReward, beneficiary); // Gets reward from yearn.

    emit ClaimedRewardFromStaking(currentEpoch, staker, stakingPositionId, beneficiary, yTokenReward, daiReward);
}
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L614-L654

```
function withdrawDaiFromVoting(uint256 votingPositionId, address voter, address beneficiary, uint256 daiNumber, bool toSwap) public only(nftVotingPosition)
{
    VotingPosition storage votingPosition = votingPositionsValues[votingPositionId];
    uint256 stakingPositionId = votingPosition.stakingPositionId;                // Gets id of staker position.
    updateInfo(stakingPositionId);                                               // Updates staking position params from previous epochs.

    require(getCurrentStage() == Stage.FirstStage || stakingPositionsValues[stakingPositionId].endDate != 0, "Wrong stage!"); // Requires correct stage or nft to be unstaked.
    require(votingPosition.endEpoch == 0, "Position removed");                   // Requires to be not liquidated yet.

    _updateVotingRewardDebt(votingPositionId);
    _subtractYTokensUserForRewardsFromVotingPosition(votingPositionId);

    if (daiNumber >= votingPosition.daiInvested)                                 // If withdraw amount more or equal of maximum invested.
    {
        _liquidateVotingPosition(votingPositionId, voter, beneficiary, stakingPositionId, toSwap);// Calls liquidate and ends call.
        return;
    }

    uint256 shares = tokensToShares(daiNumber);                                  // If withdraw amount don't require liquidating, get amount of shares and continue.

    if (toSwap == false)                                                         // If called not through swap.
    {
        vault.redeemUnderlying(shares, voter);
    }

    uint256 deltaVotes = votingPosition.daiVotes * daiNumber / votingPosition.daiInvested;// Gets average amount of votes withdrawed, cause vote price could be different.
    rewardsForEpoch[stakingPositionId][currentEpoch].yTokens -= shares;          // Decreases amount of shares for epoch.
```

```
    rewardsForEpoch[stakingPositionId][currentEpoch].votes -= deltaVotes;       // Decreases amount of votes for epoch for average votes.

    votingPosition.yTokensNumber -= _calculateVotingYTokensExcludingRewards(votingPositionId) - shares;// Decreases amount of shares.
    votingPosition.daiVotes -= deltaVotes;
    votingPosition.votes -= deltaVotes;                                         // Decreases amount of votes for position.
    votingPosition.daiInvested -= daiNumber;                                    // Decreases daiInvested amount of position.

    if (votingPosition.zooInvested > votingPosition.daiInvested)                // If zooInvested more than daiInvested left in position.
    {
        _rebalanceExceedZoo(votingPositionId, stakingPositionId, beneficiary);  // Withdraws excess zoo to save 1-1 dai-zoo proportion.
    }

    emit WithdrawedDaiFromVoting(currentEpoch, voter, stakingPositionId, beneficiary, votingPositionId, daiNumber);
}
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L662-L708

```


function _liquidateVotingPosition(uint256 votingPositionId, address voter, address beneficiary, uint256 stakingPositionId, bool toSwap) internal
{
    VotingPosition storage votingPosition = votingPositionsValues[votingPositionId];

    uint256 daiInvested = votingPosition.daiInvested;
    uint256 zooInvested = votingPosition.zooInvested;

    uint256 yTokens = votingPosition.yTokensNumber;

    if (toSwap == false)                            // If false, withdraws tokens from vault for regular liquidate.
```

```
    {
        vault.redeemUnderlying(yTokens, beneficiary);    // True when called from swapVotes, ignores withdrawal to re-assign them for another position.
    }

    _withdrawZoo(zooInvested, beneficiary); // Even if it is swap, withdraws all zoo.

    votingPosition.endEpoch = currentEpoch; // Sets endEpoch to currentEpoch.
    votingPosition.endDate = block.timestamp; // Records end date.

    rewardsForEpoch[stakingPositionId][currentEpoch].votes -= votingPosition.votes;// Decreases votes for staking position in current epoch.

    if (rewardsForEpoch[stakingPositionId][currentEpoch].yTokens >= yTokens) // If withdraws less than in staking position.
    {
        rewardsForEpoch[stakingPositionId][currentEpoch].yTokens -= yTokens; // Decreases yTokens for this staking position.
    }
    else
    {
        rewardsForEpoch[stakingPositionId][currentEpoch].yTokens = 0; // Or nullify it if trying to withdraw more yTokens than left in position(because of yTokens current rate)
    }

    // IF there is votes on position AND staking position is active
    if (rewardsForEpoch[stakingPositionId][currentEpoch].votes == 0 && stakingPositionsValues[stakingPositionId].endDate == 0)
    {
        // Move staking position to part, where staked without votes.
        for(uint256 i = 0; i < activeStakerPositions.length; i++)
        {
            if (activeStakerPositions[i] == stakingPositionId)
            {
                (activeStakerPositions[i], activeStakerPositions[numberOfNftsWithNonZeroVotes - 1]) = (activeStakerPositions[numberOfNftsWithNonZeroVotes - 1], activeStakerPositions[i]); // Swaps position to end of array
                numberOfNftsWithNonZeroVotes--; // Decrements amount of non-zero positions.
```

```
                break;
            }
        }
    }

    emit LiquidatedVotingPosition(currentEpoch, voter, stakingPositionId, beneficiary, votingPositionId, zooInvested * 995 / 1000, daiInvested);
}
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L803-L834

```
function claimRewardFromVoting(uint256 votingPositionId, address voter, address beneficiary) external only(nftVotingPosition) returns (uint256 daiReward)
{
    VotingPosition storage votingPosition = votingPositionsValues[votingPositionId];
    uint256 stakingPositionId = votingPosition.stakingPositionId; // Gets staker position id from voter position.

    require(getCurrentStage() == Stage.FirstStage || stakingPositionsValues[stakingPositionId].endDate != 0, "Wrong stage!"); // Requires to be at first stage or position should be liquidated.

    updateInfo(stakingPositionId);

    (uint256 yTokenReward, uint256 wells) = getPendingVoterReward(votingPositionId);                // Calculates amount of reward in yTokens.

    yTokenReward += votingPosition.yTokensRewardDebt; // Adds reward debt, from previous epochs.
    votingPosition.yTokensRewardDebt = 0; // Nullify reward debt.

    daiReward = vault.redeemUnderlying(yTokenReward * 980 / 1000, address(this)); // Withdraws dai from vault for yTokens, minus staker %.

    _daiRewardDistribution(beneficiary, stakingPositionId, daiReward); // Distributes reward between recipients, like treasury royalte, etc.

```

```
    if (rewardsForEpoch[stakingPositionId][currentEpoch].yTokens >= yTokenReward * 980 / 1000)
    {
        rewardsForEpoch[stakingPositionId][currentEpoch].yTokens -= yTokenReward * 980 / 1000;// Subtracts yTokens for this position.
    }
    else
    {
        rewardsForEpoch[stakingPositionId][currentEpoch].yTokens = 0;
    }

    votingPosition.lastRewardedEpoch = computeLastEpoch(votingPositionId); // Records epoch of last reward claimed.
    well.transfer(beneficiary, wells);

    emit ClaimedRewardFromVoting(currentEpoch, voter, stakingPositionId, beneficiary, yTokenReward, daiReward, votingPositionId);
}
```

`Moonwell FRAX (mFRAX)` 's redeemUnderlying function has only one parameter (`uint256 redeemAmount`) instead of two parameters (`uint256 redeemAmount, address recipient`).

The current interface definition is wrong and as a result, all the occurrences of external calls to this method will revert the whole transaction.

Status

✓ Fixed

[WP-H5] xZoo rewards can be stolen with flash loan-aided attack

High

Issue Description

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/xZoo.sol#L128-L144

```
function addZoo(uint256 positionId, uint256 amount) external
{
    require(ownerOf(positionId) == msg.sender);
    ZooStakerPosition storage position = xZooPositions[positionId];
    require(position.endEpoch == 0);
    updateTotalStakedUpdated();

    zoo.transferFrom(msg.sender, address(this), amount);

    position.yTokensDebt = getPendingReward(positionId);
    position.startEpoch = arena.currentEpoch();

    position.amount += amount;
    totalStakedZoo[arena.currentEpoch() + 1] += int256(amount);

    emit ZooStaked(msg.sender, ownerOf(positionId), amount, positionId);
}
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/xZoo.sol#L92-L106

```
function unlockZoo(uint256 positionId, address beneficiary) external returns
(uint256 amountOfZoo)
{
    require(ownerOf(positionId) == msg.sender);
    updateTotalStakedUpdated();

    ZooStakerPosition storage position = xZooPositions[positionId];
    require(position.endEpoch == 0);
    position.endEpoch = arena.currentEpoch();
    zoo.transfer(beneficiary, position.amount);
```

```
    totalStakedZoo[arena.currentEpoch() + 1] -= int256(position.amount);

    emit ZooWithdrawal(msg.sender, beneficiary, position.amount, positionId);

    return position.amount;
}
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/xZoo.sol#L172-L182

```
function getPendingReward(uint256 positionId) public view returns (uint256 yvTokens)
{
    ZooStakerPosition storage position = xZooPositions[positionId];
    uint256 end = position.endEpoch == 0 ? arena.currentEpoch() : position.endEpoch;
    yvTokens += position.yTokensDebt;

    for (uint256 epoch = position.startEpoch; epoch < end; epoch++)
    {
        yvTokens += position.amount * arena.xZooRewards(epoch) / uint256(totalStakedZoo[epoch]);
    }
}
```

A xZooPosition will be eligible for proportional rewards to the position size as long as `position.endEpoch > position.startEpoch` .

However, an attacker can do the following in the same block to take a large portion of the rewards by the time the epoch just ended but not updated yet:

1. Take a flashloan of a huge amount of ZOO tokens;
2. `xZoo.addZoo()` ;
3. `arena.updateEpoch()` ;
4. `unlockZoo()` ;
5. Repay the flashloan.

By doing so, the attacker would be able to take the majority share of the xZoo rewards for the

current epoch without actually locking any of the ZOO tokens.

Recommendation

`addZoo()` should also be changed to be like `stakeZoo()` and the `startEpoch` should be updated to `arena.currentEpoch() + 1`.

See: https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/xZoo.sol#L63-L75

Status

✓ **Fixed**

[WP-H6] Only 1/10 of the WELL rewards can be claimed with `claimRewardFromVoting()`

High

Issue Description

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L855-L878

```
function getPendingVoterReward(uint256 votingPositionId) public view returns (uint256 yTokens, uint256 wells)
{
    VotingPosition storage votingPosition = votingPositionsValues[votingPositionId];

    uint256 startEpoch = votingPosition.lastRewardedEpoch;
    uint256 endEpoch = computeLastEpoch(votingPositionId);

    uint256 stakingPositionId = votingPosition.stakingPositionId; // Gets staker position id from voter position.

    for (uint256 i = startEpoch; i < endEpoch; i++)
    {
        int256 saldo = rewardsForEpoch[stakingPositionId][i].yTokensSaldo; // Gets saldo from staker position for every epoch in range.

        uint256 totalVotes = rewardsForEpoch[stakingPositionId][i].votes;      // Gets total votes from staker position.
        if (saldo > 0)
        {

            yTokens += uint256(saldo) * votingPosition.votes / totalVotes; // Calculates yTokens amount for voter.
            wells += wellClaimedByEpoch[i] * votingPosition.votes / totalVotes / 10;
        }
    }

    return (yTokens, wells);
}
```

Seems like there is no way to get back the rest 9/10 of the WELL rewards.

Status

✓ Fixed

[WP-H7] `winnerRewards.yTokens` already includes `income1`

High

Issue Description

L1028

`uint256 income1 = winnerRewards.yTokens - tokensToShares(winnerRewards.tokensAtBattleStart);`

`income1` is calculated by comparing the current `yTokens` amount with `tokensToShares(winnerRewards.tokensAtBattleStart)`, which means `winnerRewards.yTokens` already includes `income1`.

Therefore, at L1042

`rewardsForEpoch[winner][currentEpoch + 1].yTokens = winnerRewards.yTokens + income1 + income2 - xRewar`

is double counting `income1` into the new `yTokens` amount.

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L1011-L1044

```
1011 function _calculateBattleRewards(uint256 winner, uint256 loser) internal
1012 {
1013     BattleRewardForEpoch storage winnerRewards = rewardsForEpoch[winner][currentEpoch];
1014     BattleRewardForEpoch storage loserRewards = rewardsForEpoch[loser][currentEpoch];
1015
1016     uint256 pps1 = winnerRewards.pricePerShareAtBattleStart;
1017
1018     // Skip if price per share didn't change since pairing
1019     if (pps1 == vault.exchangeRateStored())
1020     {
1021         return;
1022     }
1023
1024     winnerRewards.pricePerShareCoef = vault.exchangeRateStored() * pps1 / (vault.exchangeRateStored() - pps1);
1025     loserRewards.pricePerShareCoef = winnerRewards.pricePerShareCoef;
1026
```

```
    // Income = yTokens at battle end - yTokens at battle start
    uint256 income1 = winnerRewards.yTokens - tokensToShares(winnerRewards.tokensAtBattleStart);
    uint256 income2 = loserRewards.yTokens - tokensToShares(loserRewards.tokensAtBattleStart);

    uint256 xRewards = (income1 + income2) * 5 / 1000;
    uint256 jackpotRewards = (income1 + income2) * 1 / 100;
    vault.transfer(xZoo, xRewards); // todo: need to check that all is correct after that
    vault.transfer(jackpotA, jackpotRewards);
    vault.transfer(jackpotB, jackpotRewards);
    xZooRewards[currentEpoch] += xRewards;
    jackpotRewardsAtEpoch[currentEpoch] += jackpotRewards;
    winnerRewards.yTokensSaldo += int256(income1 + income2 - xRewards - 2 * jackpotRewards);
    loserRewards.yTokensSaldo -= int256(income2);


    rewardsForEpoch[winner][currentEpoch + 1].yTokens = winnerRewards.yTokens + income1 + income2 - xRewards - 2 * jackpotRewards;
    rewardsForEpoch[loser][currentEpoch + 1].yTokens = loserRewards.yTokens - income2;
}
```

Recommendation

```
rewardsForEpoch[winner][currentEpoch + 1].yTokens = winnerRewards.yTokens + income2 - xRewards - 2 * jackpotRewards;
```

Status

✓ Fixed

[WP-L8] `recomputeZooVotes()` doesn't make sense

Low

Issue Description

Once `recomputeDaiVotes()` is called at stage 2:

`votingPosition.daiVotes = votingPosition.votes = newVotes`

Even if `addDaiToVoting()` is called again at stage 2, `votingPosition.daiVotes = votingPosition.votes` still stands.

Thus, when `recomputeZooVotes()` is called at stage 4, at L539, `oldZooVotes` will always be `0` .

So that the check at L541 doesn't make sense.

And the computation at L543 is wrong.

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L500-L548

```
500 function recomputeDaiVotes(uint256 votingPositionId) public
501 {
502     require(getCurrentStage() == Stage.SecondStage, "Wrong stage!");
    // Requires to be at second stage of battle epoch.
503
504     VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
505
506     _updateVotingRewardDebt(votingPositionId);
507
508     uint256 stakingPositionId = votingPosition.stakingPositionId;
509     updateInfo(stakingPositionId);
    // Updates staking position params from previous epochs.
510
511     uint256 daiNumber = votingPosition.daiInvested;
    // Gets amount of dai from voting position.
512     uint256 newVotes = zooFunctions.computeVotesByDai(daiNumber);
    // Recomputes dai to votes.
```

```
    uint256 votes = votingPosition.votes; // Gets amount of votes from voting position.

    require(newVotes > votes, "Recompute to lower value"); // Requires for new votes amount to be bigger than before.

    votingPosition.daiVotes = newVotes; // Records new votes amount from dai.
    votingPosition.votes = newVotes; // Records new votes amount total.

    rewardsForEpoch[stakingPositionId][currentEpoch].votes += newVotes - votes; // Increases rewards for staker position for added amount of votes in this epoch.
    emit RecomputedDaiVotes(currentEpoch, msg.sender, stakingPositionId, votingPositionId, newVotes, votes);
}

/// @notice Function to recompute votes from zoo.
/// @param votingPositionId - id of voting position.
function recomputeZooVotes(uint256 votingPositionId) public
{
    require(getCurrentStage() == Stage.FourthStage, "Wrong stage!"); // Requires to be at 4th stage.

    VotingPosition storage votingPosition = votingPositionsValues[votingPositionId];

    _updateVotingRewardDebt(votingPositionId);

    uint256 stakingPositionId = votingPosition.stakingPositionId;
    updateInfo(stakingPositionId);

    uint256 zooNumber = votingPosition.zooInvested; // Gets amount of zoo invested from voting position.
    uint256 newZooVotes = zooFunctions.computeVotesByZoo(zooNumber); // Recomputes zoo to votes.
    uint256 oldZooVotes = votingPosition.votes - votingPosition.daiVotes;

    require(newZooVotes > oldZooVotes, "Recompute to lower value"); // Requires for new votes amount to be bigger than before.

    uint256 delta = newZooVotes + votingPosition.daiVotes / votingPosition.votes; // Gets amount of recently added zoo votes.
```

```
    rewardsForEpoch[stakingPositionId][currentEpoch].votes += delta; // Adds amount of recently added votes to reward for staker position for current epoch.
    votingPosition.votes += delta; // Add amount of recently added votes to total votes in voting position.

    emit RecomputedZooVotes(currentEpoch, msg.sender, stakingPositionId, votingPositionId, newZooVotes, oldZooVotes);
}
```

Recommendation

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L526-L548

```
function recomputeZooVotes(uint256 votingPositionId) public
{
    require(getCurrentStage() == Stage.FourthStage, "Wrong stage!"); // Requires to be at 4th stage.

    VotingPosition storage votingPosition = votingPositionsValues[votingPositionId];

    _updateVotingRewardDebt(votingPositionId);

    uint256 stakingPositionId = votingPosition.stakingPositionId;
    updateInfo(stakingPositionId);

    uint256 zooNumber = votingPosition.zooInvested; // Gets amount of zoo invested from voting position.
    uint256 newZooVotes = zooFunctions.computeVotesByZoo(zooNumber); // Recomputes zoo to votes.
    uint256 oldZooVotes = votingPosition.votes - votingPosition.daiVotes;

    require(newZooVotes > oldZooVotes, "Recompute to lower value"); // Requires for new votes amount to be bigger than before.

    uint256 delta = newZooVotes - oldZooVotes; // Gets amount of recently added zoo votes.
    rewardsForEpoch[stakingPositionId][currentEpoch].votes += delta; // Adds amount of recently added votes to reward for staker position for current epoch.
```

```
    votingPosition.votes += delta;
// Add amount of recently added votes to total votes in voting position.

    emit RecomputedZooVotes(currentEpoch, msg.sender, stakingPositionId,
votingPositionId, newZooVotes, oldZooVotes);
}
```

Status

✓ Fixed

[WP-L9] When the result of `zooFunctions.getRandomResultByEpoch()` is 0, the `WinnerChoosed` event should not be emitted

Low

Issue Description

When `random == 0` it still emits `WinnerChoosed` while it actually hasn't been chosen:

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/Jackpot.sol#L16

```
uint256 public positionIndex = 1;
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/Jackpot.sol#L62-L84

```
function chooseWinner(uint256 epoch) external
{
    require(epoch < arena.currentEpoch(), "only played epochs");
    require(winners[epoch] == 0, "winner has not choosen");
    uint256 random = zooFunctions.getRandomResultByEpoch(epoch) % map.size();
    winners[epoch] = map.get(map.getKeyAtIndex(random));

    emit WinnerChoosed(epoch, winners[epoch]);
}

function stake(uint256 id, address beneficiary) external returns (uint256)
{
    require(tokenOfOwner[msg.sender] == 0, "Caller must have only one jackpot position");

    positionContract.transferFrom(msg.sender, address(this), id);
    _mint(beneficiary, positionIndex);
    stakedPositionsById[positionIndex] = id;
    map.set(positionIndex, positionIndex);
    tokenOfOwner[msg.sender] = positionIndex;
```

```

    emit Staked(id, msg.sender, beneficiary, positionIndex);
    return positionIndex++;
}
```

Recommendation

```
function chooseWinner(uint256 epoch) external
{
    require(epoch < arena.currentEpoch(), "only played epochs");
    require(winners[epoch] == 0, "winner has not choosen");
    uint256 random = zooFunctions.getRandomResultByEpoch(epoch) % map.size();
    require(random != 0, "winner has not choosen")
    winners[epoch] = map.get(map.getKeyAtIndex(random));

    emit WinnerChoosed(epoch, winners[epoch]);
}
```

Status

✓ **Fixed**

[WP-L10] The condition expression to detect if `stakingPositionId` is the first item with ZeroVotes in `activeStakerPositions` is wrong

Low

Issue Description

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L391-L433

```
function _createVotingPosition(uint256 stakingPositionId, address voter, uint256 yTokens, uint256 amount) internal returns (uint256 votes, uint256 votingPositionId)
{
@@ 393,406 @@
    BattleRewardForEpoch storage battleReward = rewardsForEpoch[stakingPositionId][currentEpoch];

    if (battleReward.votes == 0) // If staker position had zero votes before,
    {
        for(uint256 i = 0; i < activeStakerPositions.length; i++) // Iterate for active staker positions.
        {
            if (activeStakerPositions[i] == stakingPositionId) // Finds this position.
            {
                if (stakingPositionId != numberOfNftsWithNonZeroVotes) // if equal, then its already in needed place in array.
                {
                    (activeStakerPositions[i], activeStakerPositions[numberOfNftsWithNonZeroVotes]) = (activeStakerPositions[numberOfNftsWithNonZeroVotes], activeStakerPositions[i]);// Swaps this position in array, moving it to last point of non-zero positions.
                }
                numberOfNftsWithNonZeroVotes++; // Increases amount of nft eligible for pairing.
                break;
            }
```

```
        }
    }

    battleReward.votes += votes;
// Adds votes for staker position for this epoch.
    battleReward.yTokens += yTokens;
// Adds yTokens for this staker position for this epoch.
@@ 427,431 @@
}
```

Recommendation

Consider chainging to:

```
function _createVotingPosition(uint256 stakingPositionId, address voter, uint256
yTokens, uint256 amount) internal returns (uint256 votes, uint256
votingPositionId)
{
@@ 393,406 @@
    BattleRewardForEpoch storage battleReward =
rewardsForEpoch[stakingPositionId][currentEpoch];

    if (battleReward.votes == 0)
// If staker position had zero votes before,
    {
        for(uint256 i = 0; i < activeStakerPositions.length; i++)
// Iterate for active staker positions.
        {
            if (activeStakerPositions[i] == stakingPositionId)
// Finds this position.
            {
                if (i > numberOfNftsWithNonZeroVotes)                    // if
equal, then its already in needed place in array.
                {
                    (activeStakerPositions[i],
activeStakerPositions[numberOfNftsWithNonZeroVotes]) =
(activeStakerPositions[numberOfNftsWithNonZeroVotes], activeStakerPositions[i]);//
Swaps this position in array, moving it to last point of non-zero positions.
                }
```

```
                numberOfNftsWithNonZeroVotes++;
// Increases amount of nft eligible for pairing.
                break;
            }
        }
    }

    battleReward.votes += votes;
// Adds votes for staker position for this epoch.
    battleReward.yTokens += yTokens;
// Adds yTokens for this staker position for this epoch.
```

@@ 427,431 @@

```
}
```

Status

✓ Fixed

[WP-L11] Missing return value

Low

Issue Description

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftVotingPosition.sol#L49-L53

```
49 function addDaiToPosition(uint256 votingPositionId, uint256 amount) external onlyVotingOwner(votingPositionId) returns (uint256 votes)
50 {
51     dai.transferFrom(msg.sender, address(nftBattleArena), amount); // Transfers DAI to arena contract for vote.
52     nftBattleArena.addDaiToVoting(votingPositionId, msg.sender, amount, 0); // zero for yTokens coz its not swap.
53 }
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftVotingPosition.sol#L55-L59

```
55 function addZooToPosition(uint256 votingPositionId, uint256 amount) external onlyVotingOwner(votingPositionId) returns (uint256 votes)
56 {
57     zoo.transferFrom(msg.sender, address(nftBattleArena), amount); // Transfers ZOO to arena contract for vote.
58     nftBattleArena.addZooToVoting(votingPositionId, msg.sender, amount);
59 }
```

Recommendation

```
49 function addDaiToPosition(uint256 votingPositionId, uint256 amount) external onlyVotingOwner(votingPositionId) returns (uint256 votes)
50 {
51     dai.transferFrom(msg.sender, address(nftBattleArena), amount); // Transfers DAI to arena contract for vote.
52     return nftBattleArena.addDaiToVoting(votingPositionId, msg.sender, amount, 0); // zero for yTokens coz its not swap.
```

```
}
```

```
function addZooToPosition(uint256 votingPositionId, uint256 amount) external
onlyVotingOwner(votingPositionId) returns (uint256 votes)
{
    zoo.transferFrom(msg.sender, address(nftBattleArena), amount);
// Transfers ZOO to arena contract for vote.
    return nftBattleArena.addZooToVoting(votingPositionId, msg.sender, amount);
}
```

Status

✓ Fixed

[WP-L12] Open permissionless init functions can be front run by attackers

Low

Issue Description

The `NftBattleArena.sol#init()` , `Jackpot.sol#setNftBattleArena()` , and `xZoo.sol#setNftBattleArena()` are all open & permissionless methods that can be called by any address.

While they should only be called by the deployer, it's possible that the attacker can listen to the deployment events of the deployer's address and front run the deployer's init call.

Then it will force the deployer to redeploy the contracts or if the deployer is not careful enough, the contracts being configured with the wrong addresses might get used in production and cause bigger problems.

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/NftBattleArena.sol#L261-L268

```
function init(address _xZoo, address _jackpotA, address _jackpotB) external
{
    require(xZoo == address(0));

    xZoo = _xZoo;
    jackpotA = _jackpotA;
    jackpotB = _jackpotB;
}
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/xZoo.sol#L54-L61

```
function setNftBattleArena(address _nftBattleArena) external
{
    require(address(arena) == address(0));

    arena = NftBattleArena(_nftBattleArena);
```

```

    emit NftBattleArenaSet(_nftBattleArena);
}
```

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/Jackpot.sol#L53-L60

```
function setNftBattleArena(address _nftBattleArena) external
{
    require(address(arena) == address(0));

    arena = NftBattleArena(_nftBattleArena);

    emit NftBattleArenaSet(_nftBattleArena);
}
```

Recommendation

For `Jackpot.sol#setNftBattleArena()` and `xZoo.sol#setNftBattleArena()` , consider only allowing the deployer to call `setNftBattleArena()` :

```
address immutable deployer;

constructor (address _positionContract, address _vault, address _functions, string memory _name, string memory _symbol) ERC721(_name, _symbol)
{
    vault = VaultAPI(_vault);
    positionContract = IERC721(_positionContract);
    zooFunctions = IZooFunctions(_functions);
    deployer = msg.sender;
}

function setNftBattleArena(address _nftBattleArena) external
{
    require(deployer == msg.sender);

    arena = NftBattleArena(_nftBattleArena);

    emit NftBattleArenaSet(_nftBattleArena);
```

```
}
```

And consider moving the `NftBattleArena#init()` function to the constructor:

```
constructor (
    address _zoo,
    address _dai,
    address _vault,
    address _zooGovernance,
    address _treasuryPool,
    address _gasFeePool,
    address _teamAddress,
    address _nftStakingPosition,
    address _nftVotingPosition,
    address _veZoo,
    address _controller,
    address _well,
    address _xZoo,
    address _jackpotA,
    address _jackpotB
    )
{
    zoo = ERC20(_zoo);
    dai = ERC20(_dai);
    vault = VaultAPI(_vault);
    zooGovernance = ZooGovernance(_zooGovernance);
    zooFunctions = IZooFunctions(zooGovernance.zooFunctions());
    veZoo = ListingList(_veZoo);

    treasury = _treasuryPool;
    gasPool = _gasFeePool;
    team = _teamAddress;
    nftStakingPosition = _nftStakingPosition;
    nftVotingPosition = _nftVotingPosition;

    //battlesStartDate = block.timestamp;
    epochStartDate = block.timestamp;^^I//todo:change time for prod + n days; // Start date of 1st battle.
    epochsStarts[currentEpoch] = block.timestamp;
    tokenController = ControllerInterface(_controller);
    well = ERC20(_well);
```

```

    xZoo = _xZoo;
    jackpotA = _jackpotA;
    jackpotB = _jackpotB;
}
```

Status

ⓘ Acknowledged

[WP-N13] Wrong error message

Issue Description

https://github.com/ZooDAO-Project/moonbeam-battles/blob/c5cbb8eac31d9b39a5b11ceb109dd168c4566079/contracts/Jackpot.sol#L62-L70

```
62  function chooseWinner(uint256 epoch) external
63  {
64      require(epoch < arena.currentEpoch(), "only played epochs");
65      require(winners[epoch] == 0, "winner has not choosen");
66      uint256 random = zooFunctions.getRandomResultByEpoch(epoch) % map.size();
67      winners[epoch] = map.get(map.getKeyAtIndex(random));
68
69      emit WinnerChoosed(epoch, winners[epoch]);
70  }
```

Recommendation

```
65  require(winners[epoch] == 0, "winner has been choosen");
```

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.