



ZooDAO Moonbeam Battle Audit Report

Feb 27, 2023



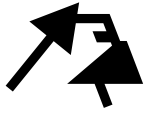


Table of Contents

Summary	2
Overview	3
Issues	4
[WP-H1] GLMR rewards are not being claimed	4
[WP-H2] <code>NftBattleArena.updateInfoAboutStakedNumber()</code> Wrong calculation of <code>lastUpdatesOfStakedNumbers</code>	6
[WP-H3] <code>updateInfo()</code> will override/discard the changes made in <code>_calculateBattleRewards()</code> because <code>lastUpdateEpoch</code> has not been changed at the same time	9
[WP-H4] Wrong interface for <code>mFRAX.redeemUnderlying()</code>	12
[WP-H5] xZoo rewards can be stolen with flash loan-aided attack	18
[WP-H6] Only 1/10 of the WELL rewards can be claimed with <code>claimRewardFromVoting()</code>	21
[WP-H7] <code>winnerRewards.yTokens</code> already includes <code>income1</code>	23
[WP-L8] <code>recomputeZooVotes()</code> doesn't make sense	25
[WP-L9] When the result of <code>zooFunctions.getRandomResultByEpoch()</code> is 0, the <code>WinnerChooosed</code> event should not be emitted	29
[WP-L10] The condition expression to detect if <code>stakingPositionId</code> is the first item with ZeroVotes in <code>activeStakerPositions</code> is wrong	31
[WP-L11] Missing return value	34
[WP-L12] Open permissionless init functions can be front run by attackers	36
[WP-N13] Wrong error message	40
Appendix	41
Disclaimer	42



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 Issues	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 implementation only claims the WELL rewards but not the GLMR rewards:

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

```
966 function requestRandom() public
967 {
968     require(getCurrentStage() == Stage.FifthStage, "Wrong stage!");
969     // Requires to be at 5th stage.
970     uint256 wellInitialBalance = well.balanceOf(address(this));
971     tokenController.claimReward(0, address(this));
972     wellClaimedByEpoch[currentEpoch] = well.balanceOf(address(this)) -
973     wellInitialBalance;
974     zooFunctions.requestRandomNumber();
975     // Calls generate random number from chainlink or blockhash.
976 }
```

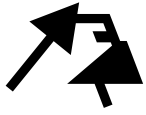
This is a big loss for all the users.

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>

```

1188 function updateInfoAboutStakedNumber(address collection) public
1189 {
1190     uint256 start = lastUpdatesOfStakedNumbers[collection] > 1 ?
        lastUpdatesOfStakedNumbers[collection] : 1;
1191     for (uint256 i = start; i <= currentEpoch; i++)
1192     {
1193         numberOfStakedNftsInCollection[i][collection] +=
        numberOfStakedNftsInCollection[i - 1][collection];
1194     }
1195
1196     lastUpdatesOfStakedNumbers[collection] = currentEpoch;
1197 }

```

```

196 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][collection]` 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][collection]`
 , 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;

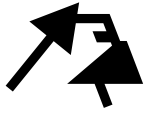
```

```
1193     uint256 start = lastUpdateEpoch > 1 ? lastUpdateEpoch : 1;
1194     for (uint256 i = start; i <= currentEpoch; i++)
1195     {
1196         numberOfStakedNftsInCollection[i][collection] +=
1197         numberOfStakedNftsInCollection[i - 1][collection];
1198     }
1199     lastUpdatesOfStakedNumbers[collection] = currentEpoch;
1200 }
```

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>

```
1048 function updateInfo(uint256 stakingPositionId) public
1049 {
1050     uint256 lastUpdateEpoch =
1051         stakingPositionsValues[stakingPositionId].lastUpdateEpoch;
1052     if (lastUpdateEpoch == currentEpoch)
1053         return;
1054     rewardsForEpoch[stakingPositionId][currentEpoch].votes =
1055         rewardsForEpoch[stakingPositionId][lastUpdateEpoch].votes;
1056     rewardsForEpoch[stakingPositionId][currentEpoch].yTokens =
1057         rewardsForEpoch[stakingPositionId][lastUpdateEpoch].yTokens;
1058     stakingPositionsValues[stakingPositionId].lastUpdateEpoch = currentEpoch;
1059 }
```

<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 =
1014         rewardsForEpoch[winner][currentEpoch];
1015     BattleRewardForEpoch storage loserRewards =
1016         rewardsForEpoch[loser][currentEpoch];
1017     uint256 pps1 = winnerRewards.pricePerShareAtBattleStart;
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
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

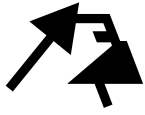
<https://moonscan.io/address/0x1C55649f73CDA2f72CEf3DD6C5CA3d49EFcF484C#writeProxyContract>

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

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

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

```
883  function claimRewardFromStaking(uint256 stakingPositionId, address staker, address
      beneficiary) public only(nftStakingPosition) returns (uint256 daiReward)
884  {
885      require(getCurrentStage() == Stage.FirstStage ||
        stakingPositionsValues[stakingPositionId].endDate != 0, "Wrong stage!"); //
        Requires to be at first stage in battle epoch.
886
887      updateInfo(stakingPositionId);
888      (uint256 yTokenReward, uint256 end) =
        getPendingStakerReward(stakingPositionId);
889      stakingPositionsValues[stakingPositionId].lastRewardedEpoch = end;
        // Records epoch of last reward claim.
890
891      daiReward = vault.redeemUnderlying(yTokenReward, beneficiary);
        // Gets reward from yearn.
892
893      emit ClaimedRewardFromStaking(currentEpoch, staker, stakingPositionId,
        beneficiary, yTokenReward, daiReward);
894  }
```



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

```
614 function withdrawDaiFromVoting(uint256 votingPositionId, address voter, address
    beneficiary, uint256 daiNumber, bool toSwap) public only(nftVotingPosition)
615 {
616     VotingPosition storage votingPosition =
        votingPositionsValues[votingPositionId];
617     uint256 stakingPositionId = votingPosition.stakingPositionId; //
        Gets id of staker position.
618     updateInfo(stakingPositionId); //
        Updates staking position params from previous epochs.
619
620     require(getCurrentStage() == Stage.FirstStage ||
        stakingPositionsValues[stakingPositionId].endDate != 0, "Wrong stage!"); //
        Requires correct stage or nft to be unstaked.
621     require(votingPosition.endEpoch == 0, "Position removed"); //
        Requires to be not liquidated yet.
622
623     _updateVotingRewardDebt(votingPositionId);
624     _subtractYTokensUserForRewardsFromVotingPosition(votingPositionId);
625
626     if (daiNumber >= votingPosition.daiInvested) //
        If withdraw amount more or equal of maximum invested.
627     {
628         _liquidateVotingPosition(votingPositionId, voter, beneficiary,
        stakingPositionId, toSwap); // Calls liquidate and ends call.
629         return;
630     }
631
632     uint256 shares = tokensToShares(daiNumber); //
        If withdraw amount don't require liquidating, get amount of shares and continue.
633
634     if (toSwap == false) //
        If called not through swap.
635     {
636         vault.redeemUnderlying(shares, voter);
637     }
638
639     uint256 deltaVotes = votingPosition.daiVotes * daiNumber /
        votingPosition.daiInvested; // Gets average amount of votes withdrawn, cause vote
        price could be different.
640     rewardsForEpoch[stakingPositionId][currentEpoch].yTokens -= shares; //
        Decreases amount of shares for epoch.
```

```

641     rewardsForEpoch[stakingPositionId][currentEpoch].votes -= deltaVotes;           //
        Decreases amount of votes for epoch for average votes.
642
643     votingPosition.yTokensNumber -=
        _calculateVotingYTokensExcludingRewards(votingPositionId) - shares; // Decreases
        amount of shares.
644     votingPosition.daiVotes -= deltaVotes;
645     votingPosition.votes -= deltaVotes;                                           //
        Decreases amount of votes for position.
646     votingPosition.daiInvested -= daiNumber;                                     //
        Decreases daiInvested amount of position.
647
648     if (votingPosition.zooInvested > votingPosition.daiInvested)                 //
        If zooInvested more than daiInvested left in position.
649     {
650         _rebalanceExceedZoo(votingPositionId, stakingPositionId, beneficiary); //
        Withdraws excess zoo to save 1-1 dai-zoo proportion.
651     }
652
653     emit WithdrawedDaiFromVoting(currentEpoch, voter, stakingPositionId,
        beneficiary, votingPositionId, daiNumber);
654 }

```

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

```

662
663
664 function _liquidateVotingPosition(uint256 votingPositionId, address voter, address
        beneficiary, uint256 stakingPositionId, bool toSwap) internal
665 {
666     VotingPosition storage votingPosition =
        votingPositionsValues[votingPositionId];
667
668     uint256 daiInvested = votingPosition.daiInvested;
669     uint256 zooInvested = votingPosition.zooInvested;
670
671     uint256 yTokens = votingPosition.yTokensNumber;
672
673     if (toSwap == false) // If false, withdraws tokens
        from vault for regular liquidate.

```

```

674     {
675         vault.redeemUnderlying(yTokens, beneficiary);    // True when called from
swapVotes, ignores withdrawal to re-assign them for another position.
676     }
677
678     _withdrawZoo(zooInvested, beneficiary);
// Even if it is swap, withdraws all zoo.
679
680     votingPosition.endEpoch = currentEpoch;
// Sets endEpoch to currentEpoch.
681     votingPosition.endDate = block.timestamp;
// Records end date.
682
683     rewardsForEpoch[stakingPositionId][currentEpoch].votes -=
votingPosition.votes;// Decreases votes for staking position in current epoch.
684
685     if (rewardsForEpoch[stakingPositionId][currentEpoch].yTokens >= yTokens)
// If withdraws less than in staking position.
686     {
687         rewardsForEpoch[stakingPositionId][currentEpoch].yTokens -= yTokens;
// Decreases yTokens for this staking position.
688     }
689     else
690     {
691         rewardsForEpoch[stakingPositionId][currentEpoch].yTokens = 0;
// Or nullify it if trying to withdraw more yTokens than left in position(because
of yTokens current rate)
692     }
693
694     // IF there is votes on position AND staking position is active
695     if (rewardsForEpoch[stakingPositionId][currentEpoch].votes == 0 &&
stakingPositionsValues[stakingPositionId].endDate == 0)
696     {
697         // Move staking position to part, where staked without votes.
698         for(uint256 i = 0; i < activeStakerPositions.length; i++)
699         {
700             if (activeStakerPositions[i] == stakingPositionId)
701             {
702                 (activeStakerPositions[i],
activeStakerPositions[numberOfNftsWithNonZeroVotes - 1]) =
(activeStakerPositions[numberOfNftsWithNonZeroVotes - 1],
activeStakerPositions[i]); // Swaps position to end of array
703                 numberOfNftsWithNonZeroVotes--;
// Decrements amount of non-zero positions.

```



```

704             break;
705         }
706     }
707 }
708
709     emit LiquidatedVotingPosition(currentEpoch, voter, stakingPositionId,
    beneficiary, votingPositionId, zooInvested * 995 / 1000, daiInvested);
710 }

```

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

```

803 function claimRewardFromVoting(uint256 votingPositionId, address voter, address
    beneficiary) external only(nftVotingPosition) returns (uint256 daiReward)
804 {
805     VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
806     uint256 stakingPositionId = votingPosition.stakingPositionId;
    // Gets staker position id from voter position.
807
808     require(getCurrentStage() == Stage.FirstStage ||
    stakingPositionsValues[stakingPositionId].endDate != 0, "Wrong stage!");
    // Requires to be at first stage or position should be liquidated.
809
810     updateInfo(stakingPositionId);
811
812     (uint256 yTokenReward, uint256 wells) =
    getPendingVoterReward(votingPositionId); // Calculates amount of
    reward in yTokens.
813
814     yTokenReward += votingPosition.yTokensRewardDebt;
    // Adds reward debt, from previous epochs.
815     votingPosition.yTokensRewardDebt = 0;
    // Nullify reward debt.
816
817     daiReward = vault.redeemUnderlying(yTokenReward * 980 / 1000, address(this));
    // Withdraws dai from vault for yTokens, minus staker %.
818
819     _daiRewardDistribution(beneficiary, stakingPositionId, daiReward);
    // Distributes reward between recipients, like treasury royaltte, etc.
820

```

```

821     if (rewardsForEpoch[stakingPositionId][currentEpoch].yTokens >= yTokenReward *
980 / 1000)
822     {
823         rewardsForEpoch[stakingPositionId][currentEpoch].yTokens -= yTokenReward *
980 / 1000; // Subtracts yTokens for this position.
824     }
825     else
826     {
827         rewardsForEpoch[stakingPositionId][currentEpoch].yTokens = 0;
828     }
829
830     votingPosition.lastRewardedEpoch = computeLastEpoch(votingPositionId);
// Records epoch of last reward claimed.
831     well.transfer(beneficiary, wells);
832
833     emit ClaimedRewardFromVoting(currentEpoch, voter, stakingPositionId,
beneficiary, yTokenReward, daiReward, votingPositionId);
834 }

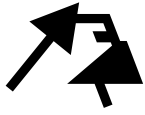
```

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>

```
128  function addZoo(uint256 positionId, uint256 amount) external
129  {
130      require(ownerOf(positionId) == msg.sender);
131      ZooStakerPosition storage position = xZooPositions[positionId];
132      require(position.endEpoch == 0);
133      updateTotalStakedUpdated();
134
135      zoo.transferFrom(msg.sender, address(this), amount);
136
137      position.yTokensDebt = getPendingReward(positionId);
138      position.startEpoch = arena.currentEpoch();
139
140      position.amount += amount;
141      totalStakedZoo[arena.currentEpoch() + 1] += int256(amount);
142
143      emit ZooStaked(msg.sender, ownerOf(positionId), amount, positionId);
144  }
```

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

```
92  function unlockZoo(uint256 positionId, address beneficiary) external returns
    (uint256 amountOfZoo)
93  {
94      require(ownerOf(positionId) == msg.sender);
95      updateTotalStakedUpdated();
96
97      ZooStakerPosition storage position = xZooPositions[positionId];
98      require(position.endEpoch == 0);
99      position.endEpoch = arena.currentEpoch();
100     zoo.transfer(beneficiary, position.amount);
```

```

101     totalStakedZoo[arena.currentEpoch() + 1] -= uint256(position.amount);
102
103     emit ZooWithdrawal(msg.sender, beneficiary, position.amount, positionId);
104
105     return position.amount;
106 }

```

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

```

172 function getPendingReward(uint256 positionId) public view returns (uint256
    yvTokens)
173 {
174     ZooStakerPosition storage position = xZooPositions[positionId];
175     uint256 end = position.endEpoch == 0 ? arena.currentEpoch() :
    position.endEpoch;
176     yvTokens += position.yTokensDebt;
177
178     for (uint256 epoch = position.startEpoch; epoch < end; epoch++)
179     {
180         yvTokens += position.amount * arena.xZooRewards(epoch) /
    uint256(totalStakedZoo[epoch]);
181     }
182 }

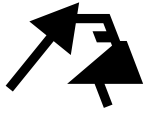
```

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

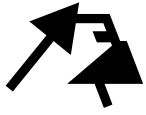
Issue Description

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

```

855  function getPendingVoterReward(uint256 votingPositionId) public view returns
      (uint256 yTokens, uint256 wells)
856  {
857      VotingPosition storage votingPosition =
      votingPositionsValues[votingPositionId];
858
859      uint256 startEpoch = votingPosition.lastRewardedEpoch;
860      uint256 endEpoch = computeLastEpoch(votingPositionId);
861
862      uint256 stakingPositionId = votingPosition.stakingPositionId;
      // Gets staker position id from voter position.
863
864      for (uint256 i = startEpoch; i < endEpoch; i++)
865      {
866          int256 saldo = rewardsForEpoch[stakingPositionId][i].yTokensSaldo;
      // Gets saldo from staker position for every epoch in range.
867
868          uint256 totalVotes = rewardsForEpoch[stakingPositionId][i].votes;      //
      Gets total votes from staker position.
869          if (saldo > 0)
870          {
871
872              yTokens += uint256(saldo) * votingPosition.votes / totalVotes;
      // Calculates yTokens amount for voter.
873              wells += wellClaimedByEpoch[i] * votingPosition.votes / totalVotes /
      10;
874          }
875      }
876
877      return (yTokens, wells);
878  }

```



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 - xRewards;
```

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 =
1014     rewardsForEpoch[winner][currentEpoch];
1014     BattleRewardForEpoch storage loserRewards =
1015     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 /
1025     (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 }

```

Recommendation

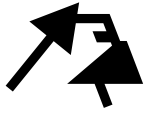
```

1042     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.

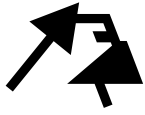
Thus, when `recomputeZooVotes()` is called at stage 4, at L539, `oldZooVotes` will always be `0`.

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

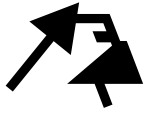
And the computation at L543 is wrong.

<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!");
503     // Requires to be at second stage of battle epoch.
504     VotingPosition storage votingPosition =
505         votingPositionsValues[votingPositionId];
506     _updateVotingRewardDebt(votingPositionId);
507
508     uint256 stakingPositionId = votingPosition.stakingPositionId;
509     updateInfo(stakingPositionId);
510     // Updates staking position params from previous epochs.
511     uint256 daiNumber = votingPosition.daiInvested;
512     // Gets amount of dai from voting position.
513     uint256 newVotes = zooFunctions.computeVotesByDai(daiNumber);
514     // Recomputes dai to votes.
```



```
513     uint256 votes = votingPosition.votes;
    // Gets amount of votes from voting position.
514
515     require(newVotes > votes, "Recompute to lower value");
    // Requires for new votes amount to be bigger than before.
516
517     votingPosition.daiVotes = newVotes;
    // Records new votes amount from dai.
518     votingPosition.votes = newVotes;
    // Records new votes amount total.
519
520     rewardsForEpoch[stakingPositionId][currentEpoch].votes += newVotes - votes;
    // Increases rewards for staker position for added amount of votes in this epoch.
521     emit RecomputedDaiVotes(currentEpoch, msg.sender, stakingPositionId,
    votingPositionId, newVotes, votes);
522 }
523
524 /// @notice Function to recompute votes from zoo.
525 /// @param votingPositionId - id of voting position.
526 function recomputeZooVotes(uint256 votingPositionId) public
527 {
528     require(getCurrentStage() == Stage.FourthStage, "Wrong stage!");
    // Requires to be at 4th stage.
529
530     VotingPosition storage votingPosition =
    votingPositionsValues[votingPositionId];
531
532     _updateVotingRewardDebt(votingPositionId);
533
534     uint256 stakingPositionId = votingPosition.stakingPositionId;
535     updateInfo(stakingPositionId);
536
537     uint256 zooNumber = votingPosition.zooInvested;
    // Gets amount of zoo invested from voting position.
538     uint256 newZooVotes = zooFunctions.computeVotesByZoo(zooNumber);
    // Recomputes zoo to votes.
539     uint256 oldZooVotes = votingPosition.votes - votingPosition.daiVotes;
540
541     require(newZooVotes > oldZooVotes, "Recompute to lower value");
    // Requires for new votes amount to be bigger than before.
542
543     uint256 delta = newZooVotes + votingPosition.daiVotes / votingPosition.votes;
    // Gets amount of recently added zoo votes.
```




```
544     rewardsForEpoch[stakingPositionId][currentEpoch].votes += delta;
      // Adds amount of recently added votes to reward for staker position for current
      epoch.
545     votingPosition.votes += delta;
      // Add amount of recently added votes to total votes in voting position.
546
547     emit RecomputedZooVotes(currentEpoch, msg.sender, stakingPositionId,
      votingPositionId, newZooVotes, oldZooVotes);
548 }
```

Recommendation

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

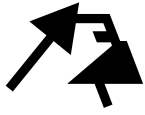
```
526  function recomputeZooVotes(uint256 votingPositionId) public
527  {
528      require(getCurrentStage() == Stage.FourthStage, "Wrong stage!");
      // Requires to be at 4th stage.
529
530      VotingPosition storage votingPosition =
      votingPositionsValues[votingPositionId];
531
532      _updateVotingRewardDebt(votingPositionId);
533
534      uint256 stakingPositionId = votingPosition.stakingPositionId;
535      updateInfo(stakingPositionId);
536
537      uint256 zooNumber = votingPosition.zooInvested;
      // Gets amount of zoo invested from voting position.
538      uint256 newZooVotes = zooFunctions.computeVotesByZoo(zooNumber);
      // Recomputes zoo to votes.
539      uint256 oldZooVotes = votingPosition.votes - votingPosition.daiVotes;
540
541      require(newZooVotes > oldZooVotes, "Recompute to lower value");
      // Requires for new votes amount to be bigger than before.
542
543      uint256 delta = newZooVotes - oldZooVotes; // Gets amount of recently added
      zoo votes.
544      rewardsForEpoch[stakingPositionId][currentEpoch].votes += delta;
      // Adds amount of recently added votes to reward for staker position for current
      epoch.
```



```
545     votingPosition.votes += delta;  
      // Add amount of recently added votes to total votes in voting position.  
546  
547     emit RecomputedZooVotes(currentEpoch, msg.sender, stakingPositionId,  
      votingPositionId, newZooVotes, oldZooVotes);  
548 }
```

Status

✓ Fixed



[WP-L9] When the result of

`zooFunctions.getRandomResultByEpoch()` is 0, the `WinnerChooosed` event should not be emitted

Low

Issue Description

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

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

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

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

```
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 WinnerChooosed(epoch, winners[epoch]);
70  }
71
72  function stake(uint256 id, address beneficiary) external returns (uint256)
73  {
74      require(tokenOfOwner[msg.sender] == 0, "Caller must have only one jackpot
75      position");
76      positionContract.transferFrom(msg.sender, address(this), id);
77      _mint(beneficiary, positionIndex);
78      stakedPositionsById[positionIndex] = id;
79      map.set(positionIndex, positionIndex);
80      tokenOfOwner[msg.sender] = positionIndex;
```

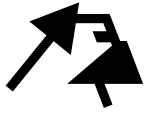
```
81
82     emit Staked(id, msg.sender, beneficiary, positionIndex);
83     return positionIndex++;
84 }
```

Recommendation

```
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     require(random != 0, "winner has not choosen")
68     winners[epoch] = map.get(map.getKeyAtIndex(random));
69
70     emit WinnerChoosed(epoch, winners[epoch]);
71 }
```

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>

```
391  function _createVotingPosition(uint256 stakingPositionId, address voter, uint256
    yTokens, uint256 amount) internal returns (uint256 votes, uint256
    votingPositionId)
392  {
    @@ 393,406 @@
407      BattleRewardForEpoch storage battleReward =
    rewardsForEpoch[stakingPositionId][currentEpoch];
408
409      if (battleReward.votes == 0)
    // If staker position had zero votes before,
410      {
411          for(uint256 i = 0; i < activeStakerPositions.length; i++)
    // Iterate for active staker positions.
412          {
413              if (activeStakerPositions[i] == stakingPositionId)
    // Finds this position.
414              {
415                  if (stakingPositionId != numberOfNftsWithNonZeroVotes)
    // if equal, then its already in needed place in array.
416                  {
417                      (activeStakerPositions[i],
    activeStakerPositions[numberOfNftsWithNonZeroVotes]) =
    (activeStakerPositions[numberOfNftsWithNonZeroVotes], activeStakerPositions[i]); //
    Swaps this position in array, moving it to last point of non-zero positions.
418                  }
419                  numberOfNftsWithNonZeroVotes++;
    // Increases amount of nft eligible for pairing.
420                  break;
421              }
```



```

422     }
423 }
424
425     battleReward.votes += votes;
    // Adds votes for staker position for this epoch.
426     battleReward.yTokens += yTokens;
    // Adds yTokens for this staker position for this epoch.
    @@ 427,431 @@
432 }

```

Recommendation

Consider chainging to:

```

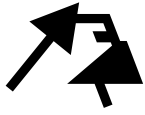
391 function _createVotingPosition(uint256 stakingPositionId, address voter, uint256
yTokens, uint256 amount) internal returns (uint256 votes, uint256
votingPositionId)
392 {
    @@ 393,406 @@
407     BattleRewardForEpoch storage battleReward =
rewardsForEpoch[stakingPositionId][currentEpoch];
408
409     if (battleReward.votes == 0)
    // If staker position had zero votes before,
410     {
411         for(uint256 i = 0; i < activeStakerPositions.length; i++)
    // Iterate for active staker positions.
412         {
413             if (activeStakerPositions[i] == stakingPositionId)
    // Finds this position.
414             {
415                 if (i > numberOfNftsWithNonZeroVotes) // if
equal, then its already in needed place in array.
416                 {
417                     (activeStakerPositions[i],
activeStakerPositions[numberOfNftsWithNonZeroVotes]) =
(activeStakerPositions[numberOfNftsWithNonZeroVotes], activeStakerPositions[i]); //
Swaps this position in array, moving it to last point of non-zero positions.
418                 }

```

```
419         numberOfNftsWithNonZeroVotes++;  
    // Increases amount of nft eligible for pairing.  
420         break;  
421     }  
422 }  
423 }  
424  
425     battleReward.votes += votes;  
    // Adds votes for staker position for this epoch.  
426     battleReward.yTokens += yTokens;  
    // Adds yTokens for this staker position for this epoch.  
@@ 427,431 @@  
432 }
```

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

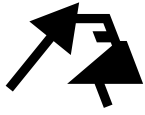
```
49  function addDaiToPosition(uint256 votingPositionId, uint256 amount) external  
    onlyVotingOwner(votingPositionId) returns (uint256 votes)  
50  {  
51      dai.transferFrom(msg.sender, address(nftBattleArena), amount);  
    // Transfers DAI to arena contract for vote.  
52      return nftBattleArena.addDaiToVoting(votingPositionId,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


53 }

```
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     return nftBattleArena.addZooToVoting(votingPositionId, msg.sender, amount);  
59 }
```

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.

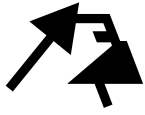
Then it will force the deployer to redeploy the contracts or if the deployer is not careful enough, the contracts being configured with the wrong addresses might get used in production and cause bigger problems.

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

```
261 function init(address _xZoo, address _jackpotA, address _jackpotB) external
262 {
263     require(xZoo == address(0));
264
265     xZoo = _xZoo;
266     jackpotA = _jackpotA;
267     jackpotB = _jackpotB;
268 }
```

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

```
54 function setNftBattleArena(address _nftBattleArena) external
55 {
56     require(address(arena) == address(0));
57
58     arena = NftBattleArena(_nftBattleArena);
```



```
59
60     emit NftBattleArenaSet(_nftBattleArena);
61 }
```

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

```
53  function setNftBattleArena(address _nftBattleArena) external
54  {
55      require(address(arena) == address(0));
56
57      arena = NftBattleArena(_nftBattleArena);
58
59      emit NftBattleArenaSet(_nftBattleArena);
60  }
```

Recommendation

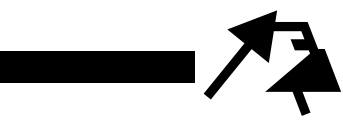
For `Jackpot.sol#setNftBattleArena()` and `xZoo.sol#setNftBattleArena()`, consider only allowing the deployer to call `setNftBattleArena()` :

```
47  address immutable deployer;
48
49  constructor (address _positionContract, address _vault, address _functions, string
memory _name, string memory _symbol) ERC721(_name, _symbol)
50  {
51      vault = VaultAPI(_vault);
52      positionContract = IERC721(_positionContract);
53      zooFunctions = IZooFunctions(_functions);
54      deployer = msg.sender;
55  }
56
57  function setNftBattleArena(address _nftBattleArena) external
58  {
59      require(deployer == msg.sender);
60
61      arena = NftBattleArena(_nftBattleArena);
62
63      emit NftBattleArenaSet(_nftBattleArena);
```

```
64 }
```

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

```
227 constructor (
228     address _zoo,
229     address _dai,
230     address _vault,
231     address _zooGovernance,
232     address _treasuryPool,
233     address _gasFeePool,
234     address _teamAddress,
235     address _nftStakingPosition,
236     address _nftVotingPosition,
237     address _veZoo,
238     address _controller,
239     address _well,
240     address _xZoo,
241     address _jackpotA,
242     address _jackpotB
243 )
244 {
245     zoo = ERC20(_zoo);
246     dai = ERC20(_dai);
247     vault = VaultAPI(_vault);
248     zooGovernance = ZooGovernance(_zooGovernance);
249     zooFunctions = IZooFunctions(zooGovernance.zooFunctions());
250     veZoo = ListingList(_veZoo);
251
252     treasury = _treasuryPool;
253     gasPool = _gasFeePool;
254     team = _teamAddress;
255     nftStakingPosition = _nftStakingPosition;
256     nftVotingPosition = _nftVotingPosition;
257
258     //battlesStartDate = block.timestamp;
259     epochStartDate = block.timestamp;^^I//todo:change time for prod + n days; //
    Start date of 1st battle.
260     epochsStarts[currentEpoch] = block.timestamp;
261     tokenController = ControllerInterface(_controller);
262     well = ERC20(_well);
```



```
263  
264     xZoo = _xZoo;  
265     jackpotA = _jackpotA;  
266     jackpotB = _jackpotB;  
267 }
```

Status

① 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.