BWARELABS Security Review

Executive Summary

The Softbinator Security team reviewed the Bware Staking Contract and have not discovered any critical/high/medium or low issues. We included in our review 4 informational level issues for the Bware Team's consideration.

Summary

Type of Project: DeFi

Timeline: Sep 5th, 2022 - Sep 12th, 2022

Methods: Manual & Automated Review

Documentation: Good

Github commit: `f99996e79b8f795d5a8d6d34bc8e4373f36e1e91`

Revision commit: `ca3fae61f36b0bc17555f74044cb0d6d4917f92b`

Second revision commit: `423650c7d2fab3758c5f798923d233af2b49ebe0`

Total Issues:

Critical/High: 0

Medium: 0

Low: 0

Gas Optimizations: 0

Informational: 4

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of Bware Staking Contract. Any modifications to the code will require a new security review

Softbinator Technologies

Softbinator Technologies is a company specialized in developing products with cutting-edge technologies. Blockchain Team has been involved in building on-chain games, Defi protocols and making security audits for various products.

Introduction

The analyzed contract is represented by a Staking Contract. It is based on Cardano Staking protocol, with the pledge mechanism, but also with the posibility to slash the nodes. This Staking contract introduce the possibility to stake multiple currencies(ERC20 tokens) into a pool(only the owner of the pool, regular delegators can stake only main token), and earn rewards. The rewards are calculated with the help of epochs and are distributed as the main token.

Epoch mechanism:

Epochs are flexible with their length, their ending being trigger by calling endEpoch, but not before a minimum duration specified in config mapping and checking that all pools are finalized. At the end of an epoch, a snapshot is made to save all the stats for the pools, like: reward amount, number of pools, configs for pledge and saturation, the length of the ended epoch, the operative amount of tokens until now over all pools, etc…

Rewards:

Rewards are calculated at the end of an epoch, and are calculated for all pools and stored in the snapshot. Then, before ending the current epoch, at the finalization of the pools, rewards are calculated for each pool and ratio rate of reward per token is stored.

Pools:

The creation of pools is restricted to a special role. A pool receives rewards starting with the next epoch.

Delegators:

Delegators (regular stakers in a pool), can stake only the main token.

INFORMATIONAL ISSUES

1. `function delegate(MoveBalanceRequest memory mbr_)`: CEI not respected

```
function delegate(MoveBalanceRequest memory mbr_) external
payPoolOpFee(mbr_.poolId) {
        mbr_.member = _msgSender();
        _strictTransferFrom(mbr_); // TODO transfer to contract
        _delegate(mbr_); // increment in contract their balances
        require(
            (mbr_.currencies.length == 1 && mbr_.currencies[0].erc20 ==
rewardsToken) ||
                _isPoolOwner(mbr_.member, mbr_.poolId),
            "delegate: delegators can only deposit main token"
        );

        emit Delegate(mbr_.member, mbr_.poolId, mbr_.currencies);
    }
```

Recommendation: Softbinator recommends moving require filed before the external call:

```
        ...
        require(
            (mbr_.currencies.length == 1 && mbr_.currencies[0].erc20 ==
rewardsToken) ||
                _isPoolOwner(mbr_.member, mbr_.poolId),
            "delegate: delegators can only deposit main token"
            );
        _strictTransferFrom(mbr_);
        ...
```

Resolution: Resolved

2. Private/Internal variables can not be viewed in production

Examples:

```
mapping(bytes32 => uint256) private lastAllocatedCRwd;
mapping(bytes32 => uint256) internal lastStoredCRwd;
```

Recommendation: Softbinator recommends creating for each private/internal variable a getter method in order to let 3rd parties to analyze them easily

Resolution: Resolved

3. `function unjail(bytes32 poolId)`: CEI not respected

```
function unjail(bytes32 poolId) external payPoolOpFee(poolId) {
        require(isPoolJailed(poolId), "unjail: pool is not jailed");

        // collect the unjailing fee
        depositFunds(getConfig(POOL_UNJAIL_FEE));
        poolsArchive[poolId].jailedToEpoch = currentEpoch;

        emit UnjailPool(poolId);
    }
```

Recommendation: Softbinator recommends to move **poolsArchive[poolId].jailedToEpoch = currentEpoch;** before **depositFunds**, which is an external call

Resolution: Resolved

4. `function _poolNotSuspended(bytes32 poolId)` is 2 out of 3 times used to determine if a pool is suspended

```
  function _poolNotSuspended(bytes32 poolId) private view returns (bool) {
        return isPoolOperative(poolId) && !isPoolJailed(poolId);
    }
```

Recommendation: Softbinator recommends to change it to return true if pool is suspended

Resolution: Resolved

Automated Review - Slither

We exclude a few detectors which weren't relevant for the contracts:

naming-convention,solc-version, pragma, timestamp, uninitialized-local

Below is the generated report, from which we excluded the analysis of Openzepellin contracts regarding dead-code.

```
$ slither ./contracts/Staking.sol --solc-remaps
'@openzeppelin=node_modules/@openzeppelin hardhat=node_modules/hardhat'
--exclude naming-convention,solc-version,pragma,timestamp,uninitialized-local
Compilation warnings/errors on ./contracts/Staking.sol:
Warning: Contract code size is 42832 bytes and exceeds 24576 bytes (a limit
introduced in Spurious Dragon). This contract may not be deployable on
Mainnet. Consider enabling the optimizer (with a low "runs" value!), turning
off revert strings, or using libraries.
 --> contracts/Staking.sol:7:1:
  |
7 | contract Staking is IPoolCreator, DelegationsAdapter {
  | ^ (Relevant source part starts here and spans across multiple lines).



ERC20Utils.strictTransferFrom(IERC20Upgradeable,address,address,uint256)
(contracts/deposits/ERC20Utils.sol#13-22) uses arbitrary from in
transferFrom: token.safeTransferFrom(from,to,value)
(contracts/deposits/ERC20Utils.sol#20)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-
in-transferfrom

Reentrancy in
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/Staking.sol#209-230):
        External calls:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#219)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - token.safeTransferFrom(from,to,value)
(contracts/deposits/ERC20Utils.sol#20)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/deposits/TransfersAdapter.sol#68)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#219)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        State variables written after the call(s):
        - _moveMemberBalance(mbr_,MoveBalanceDirection.INTO)
(contracts/Staking.sol#220)
                - delegationObj.currencies[ca_.erc20] =
changeCurrencyFn(delegationObj.currencies[ca_.erc20],ca_.amount)
(contracts/deposits/DelegationsAdapter.sol#52)
        StakingStorage.poolsArchive (contracts/StakingStorage.sol#53) can be
used in cross function reentrancies:
```

```
        - Staking._cancelWithdrawRequest(address,bytes32,bytes32)
(contracts/Staking.sol#461-473)
        -
DelegationsAdapter._changeCurrencyAmount(IMoveBalanceStructs.MoveBalanceReque
st,function(uint256,uint256) returns(uint256))
(contracts/deposits/DelegationsAdapter.sol#35-54)
        -
DelegationsAdapter._changeMemberBalance(IMoveBalanceStructs.MoveBalanceReques
t,function(DepositsAdapter.DeferredDeposit,uint256) returns(uint256))
(contracts/deposits/DelegationsAdapter.sol#61-87)
        - Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#209-258)
        - Staking._isPoolOwner(address,bytes32)
(contracts/Staking.sol#243-245)
        - Staking._setPoolBeneficiary(bytes32,address)
(contracts/Staking.sol#294-299)
        - Staking._setPoolCommission(bytes32,uint16)
(contracts/Staking.sol#266-272)
        - MemberRewardsAdapter._syncMemberRewards(address,bytes32)
(contracts/rewards/MemberRewardsAdapter.sol#212-229)
        - MemberRewardsAdapter.computeMemberReward(address,bytes32[])
(contracts/rewards/MemberRewardsAdapter.sol#144-152)
        -
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/Staking.sol#209-230)
        - Staking.executeWithdrawRequest(bytes32)
(contracts/Staking.sol#501-516)
        - Staking.getMemberBalance(address,bytes32)
(contracts/Staking.sol#174-176)
        - Staking.getMemberCurrency(address,bytes32,IERC20Upgradeable)
(contracts/Staking.sol#184-190)
        - Staking.getMemberWithdrawRequest(address,bytes32)
(contracts/Staking.sol#197-199)
        - Staking.isPoolJailed(bytes32) (contracts/Staking.sol#314-316)
        - Staking.isPoolOperative(bytes32) (contracts/Staking.sol#306-308)
        - Staking.jail(bytes32) (contracts/Staking.sol#419-424)
        - StakingStorage.poolsArchive (contracts/StakingStorage.sol#53)
        - Staking.setPoolBeneficiary(bytes32,address)
(contracts/Staking.sol#279-287)
        - Staking.setPoolCommission(bytes32,uint16)
(contracts/Staking.sol#253-259)
        -
MemberRewardsAdapter.syncMemberRewards(bytes32[],address,Finalizer.RewardSrc)
(contracts/rewards/MemberRewardsAdapter.sol#238-250)
        - Staking.undelegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/Staking.sol#371-394)
        - Staking.unjail(bytes32) (contracts/Staking.sol#431-439)
        - _setPoolBeneficiary(mbr_.poolId,beneficiary)
(contracts/Staking.sol#228)
                - poolsArchive[poolId].beneficiary = beneficiary
(contracts/Staking.sol#296)
        StakingStorage.poolsArchive (contracts/StakingStorage.sol#53) can be
used in cross function reentrancies:
```

```
        - Staking._cancelWithdrawRequest(address,bytes32,bytes32)
(contracts/Staking.sol#461-473)
        -
DelegationsAdapter._changeCurrencyAmount(IMoveBalanceStructs.MoveBalanceReque
st,function(uint256,uint256) returns(uint256))
(contracts/deposits/DelegationsAdapter.sol#35-54)
        -
DelegationsAdapter._changeMemberBalance(IMoveBalanceStructs.MoveBalanceReques
t,function(DepositsAdapter.DeferredDeposit,uint256) returns(uint256))
(contracts/deposits/DelegationsAdapter.sol#61-87)
        - Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#209-258)
        - Staking._isPoolOwner(address,bytes32)
(contracts/Staking.sol#243-245)
        - Staking._setPoolBeneficiary(bytes32,address)
(contracts/Staking.sol#294-299)
        - Staking._setPoolCommission(bytes32,uint16)
(contracts/Staking.sol#266-272)
        - MemberRewardsAdapter._syncMemberRewards(address,bytes32)
(contracts/rewards/MemberRewardsAdapter.sol#212-229)
        - MemberRewardsAdapter.computeMemberReward(address,bytes32[])
(contracts/rewards/MemberRewardsAdapter.sol#144-152)
        -
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/Staking.sol#209-230)
        - Staking.executeWithdrawRequest(bytes32)
(contracts/Staking.sol#501-516)
        - Staking.getMemberBalance(address,bytes32)
(contracts/Staking.sol#174-176)
        - Staking.getMemberCurrency(address,bytes32,IERC20Upgradeable)
(contracts/Staking.sol#184-190)
        - Staking.getMemberWithdrawRequest(address,bytes32)
(contracts/Staking.sol#197-199)
        - Staking.isPoolJailed(bytes32) (contracts/Staking.sol#314-316)
        - Staking.isPoolOperative(bytes32) (contracts/Staking.sol#306-308)
        - Staking.jail(bytes32) (contracts/Staking.sol#419-424)
        - StakingStorage.poolsArchive (contracts/StakingStorage.sol#53)
        - Staking.setPoolBeneficiary(bytes32,address)
(contracts/Staking.sol#279-287)
        - Staking.setPoolCommission(bytes32,uint16)
(contracts/Staking.sol#253-259)
        -
MemberRewardsAdapter.syncMemberRewards(bytes32[],address,Finalizer.RewardSrc)
(contracts/rewards/MemberRewardsAdapter.sol#238-250)
        - Staking.undelegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/Staking.sol#371-394)
        - Staking.unjail(bytes32) (contracts/Staking.sol#431-439)
        - _setPoolCommission(mbr_.poolId,commissionFee)
(contracts/Staking.sol#229)
                - poolsArchive[poolId].commissionFee = commission
(contracts/Staking.sol#268)
                - poolsArchive[poolId].commissionFeeEpochSet = currentEpoch
(contracts/Staking.sol#269)
```

```
        StakingStorage.poolsArchive (contracts/StakingStorage.sol#53) can be
used in cross function reentrancies:
        - Staking._cancelWithdrawRequest(address,bytes32,bytes32)
(contracts/Staking.sol#461-473)
        -
DelegationsAdapter._changeCurrencyAmount(IMoveBalanceStructs.MoveBalanceReque
st,function(uint256,uint256) returns(uint256))
(contracts/deposits/DelegationsAdapter.sol#35-54)
        -
DelegationsAdapter._changeMemberBalance(IMoveBalanceStructs.MoveBalanceReques
t,function(DepositsAdapter.DeferredDeposit,uint256) returns(uint256))
(contracts/deposits/DelegationsAdapter.sol#61-87)
        - Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#209-258)
        - Staking._isPoolOwner(address,bytes32)
(contracts/Staking.sol#243-245)
        - Staking._setPoolBeneficiary(bytes32,address)
(contracts/Staking.sol#294-299)
        - Staking._setPoolCommission(bytes32,uint16)
(contracts/Staking.sol#266-272)
        - MemberRewardsAdapter._syncMemberRewards(address,bytes32)
(contracts/rewards/MemberRewardsAdapter.sol#212-229)
        - MemberRewardsAdapter.computeMemberReward(address,bytes32[])
(contracts/rewards/MemberRewardsAdapter.sol#144-152)
        -
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/Staking.sol#209-230)
        - Staking.executeWithdrawRequest(bytes32)
(contracts/Staking.sol#501-516)
        - Staking.getMemberBalance(address,bytes32)
(contracts/Staking.sol#174-176)
        - Staking.getMemberCurrency(address,bytes32,IERC20Upgradeable)
(contracts/Staking.sol#184-190)
        - Staking.getMemberWithdrawRequest(address,bytes32)
(contracts/Staking.sol#197-199)
        - Staking.isPoolJailed(bytes32) (contracts/Staking.sol#314-316)
        - Staking.isPoolOperative(bytes32) (contracts/Staking.sol#306-308)
        - Staking.jail(bytes32) (contracts/Staking.sol#419-424)
        - StakingStorage.poolsArchive (contracts/StakingStorage.sol#53)
        - Staking.setPoolBeneficiary(bytes32,address)
(contracts/Staking.sol#279-287)
        - Staking.setPoolCommission(bytes32,uint16)
(contracts/Staking.sol#253-259)
        -
MemberRewardsAdapter.syncMemberRewards(bytes32[],address,Finalizer.RewardSrc)
(contracts/rewards/MemberRewardsAdapter.sol#238-250)
        - Staking.undelegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/Staking.sol#371-394)
        - Staking.unjail(bytes32) (contracts/Staking.sol#431-439)
Reentrancy in Staking.unjail(bytes32) (contracts/Staking.sol#431-439):
        External calls:
        - depositFunds(getConfig(POOL_UNJAIL_FEE))
(contracts/Staking.sol#435)
```

```
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - token.safeTransferFrom(from,to,value)
(contracts/deposits/ERC20Utils.sol#20)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/deposits/TransfersAdapter.sol#68)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - depositFunds(getConfig(POOL_UNJAIL_FEE))
(contracts/Staking.sol#435)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        State variables written after the call(s):
        - poolsArchive[poolId].jailedToEpoch = currentEpoch
(contracts/Staking.sol#436)
        StakingStorage.poolsArchive (contracts/StakingStorage.sol#53) can be
used in cross function reentrancies:
        - Staking._cancelWithdrawRequest(address,bytes32,bytes32)
(contracts/Staking.sol#461-473)
        -
DelegationsAdapter._changeCurrencyAmount(IMoveBalanceStructs.MoveBalanceReque
st,function(uint256,uint256) returns(uint256))
(contracts/deposits/DelegationsAdapter.sol#35-54)
        -
DelegationsAdapter._changeMemberBalance(IMoveBalanceStructs.MoveBalanceReques
t,function(DepositsAdapter.DeferredDeposit,uint256) returns(uint256))
(contracts/deposits/DelegationsAdapter.sol#61-87)
        - Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#209-258)
        - Staking._isPoolOwner(address,bytes32)
(contracts/Staking.sol#243-245)
        - Staking._setPoolBeneficiary(bytes32,address)
(contracts/Staking.sol#294-299)
        - Staking._setPoolCommission(bytes32,uint16)
(contracts/Staking.sol#266-272)
        - MemberRewardsAdapter._syncMemberRewards(address,bytes32)
(contracts/rewards/MemberRewardsAdapter.sol#212-229)
        - MemberRewardsAdapter.computeMemberReward(address,bytes32[])
(contracts/rewards/MemberRewardsAdapter.sol#144-152)
        -
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/Staking.sol#209-230)
        - Staking.executeWithdrawRequest(bytes32)
(contracts/Staking.sol#501-516)
        - Staking.getMemberBalance(address,bytes32)
(contracts/Staking.sol#174-176)
```

```
        - Staking.getMemberCurrency(address,bytes32,IERC20Upgradeable)
(contracts/Staking.sol#184-190)
        - Staking.getMemberWithdrawRequest(address,bytes32)
(contracts/Staking.sol#197-199)
        - Staking.isPoolJailed(bytes32) (contracts/Staking.sol#314-316)
        - Staking.isPoolOperative(bytes32) (contracts/Staking.sol#306-308)
        - Staking.jail(bytes32) (contracts/Staking.sol#419-424)
        - StakingStorage.poolsArchive (contracts/StakingStorage.sol#53)
        - Staking.setPoolBeneficiary(bytes32,address)
(contracts/Staking.sol#279-287)
        - Staking.setPoolCommission(bytes32,uint16)
(contracts/Staking.sol#253-259)
        -
MemberRewardsAdapter.syncMemberRewards(bytes32[],address,Finalizer.RewardSrc)
(contracts/rewards/MemberRewardsAdapter.sol#238-250)
        - Staking.undelegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/Staking.sol#371-394)
        - Staking.unjail(bytes32) (contracts/Staking.sol#431-439)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vuln
erabilities

Finalizer._computeEpochRewards(Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#126-130) performs a multiplication on the
result of a division:
        - epochRewards = (snapshot.depositedTotal *
getConfig(AVERAGE_TOKEN_APY)) / MAX_PPM (contracts/rewards/Finalizer.sol#127)
        - epochRewards = (epochRewards * snapshot.prevEpochLength) /
ONE_YEAR_SEC (contracts/rewards/Finalizer.sol#128)
Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#209-258) performs a multiplication on the
result of a division:
        - chainReward = (_getChainRewards(poolId) * snapshot.prevEpochLength)
/ ONE_YEAR_SEC (contracts/rewards/Finalizer.sol#231)
        - chainReward = (chainReward * performance) / MAX_PPM
(contracts/rewards/Finalizer.sol#235)
Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#209-258) performs a multiplication on the
result of a division:
        - membersReward = (membersReward * performance) / MAX_PPM
(contracts/rewards/Finalizer.sol#236)
        - ownerReward = (membersReward * poolObj.commissionFee) / MAX_PPM
(contracts/rewards/Finalizer.sol#244)
Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#209-258) performs a multiplication on the
result of a division:
        - membersReward = (membersReward * performance) / MAX_PPM
(contracts/rewards/Finalizer.sol#236)
        - cr_.rewardRatio =
poolCumulativeReward[poolId][snapshot.finalizerEpoch - 1].rewardRatio +
SafeCastUpgradeable.toUint128((membersReward * ONE_TOKEN_UNITS) / delegated)
(contracts/rewards/Finalizer.sol#252-254)
```

```
Finalizer._computePoolEpochRewards(uint256,uint256,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#267-283) performs a multiplication on the
result of a division:
        - poolRewards = (pledged * (delegated - poolRewards)) /
snapshot.saturationCap (contracts/rewards/Finalizer.sol#276)
        - poolRewards = (poolRewards * snapshot.pledgeFactorNum) /
snapshot.pledgeFactorDenom (contracts/rewards/Finalizer.sol#277)
Finalizer._computePoolEpochRewards(uint256,uint256,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#267-283) performs a multiplication on the
result of a division:
        - poolRewards = (poolRewards * snapshot.pledgeFactorNum) /
snapshot.pledgeFactorDenom (contracts/rewards/Finalizer.sol#277)
        - poolRewards = (poolRewards * (snapshot.pledgeFactorDenom)) /
(snapshot.pledgeFactorDenom + snapshot.pledgeFactorNum)
(contracts/rewards/Finalizer.sol#279-281)
Finalizer._computePoolEpochRewards(uint256,uint256,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#267-283) performs a multiplication on the
result of a division:
        - poolRewards = (poolRewards * (snapshot.pledgeFactorDenom)) /
(snapshot.pledgeFactorDenom + snapshot.pledgeFactorNum)
(contracts/rewards/Finalizer.sol#279-281)
        - poolRewards = (snapshot.prevEpochRewards * poolRewards) /
snapshot.depositedTotal (contracts/rewards/Finalizer.sol#282)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse = (3 * denominator) ^ 2
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
```

```
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - prod0 = prod0 / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#105)
        - result = prod0 * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#132)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-m
ultiply

MemberRewardsAdapter._computeMemberRewardOverInterval(bytes32,uint256,uint64,
uint64).endEpoch (contracts/rewards/MemberRewardsAdapter.sol#123) shadows:
        - Finalizer.endEpoch() (contracts/rewards/Finalizer.sol#136-165)
(function)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing
```

```
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#128-137) has external calls inside a loop: (success,returndata) =
target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a
-loop

Reentrancy in
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/Staking.sol#209-230):
        External calls:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#219)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - token.safeTransferFrom(from,to,value)
(contracts/deposits/ERC20Utils.sol#20)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/deposits/TransfersAdapter.sol#68)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#219)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        State variables written after the call(s):
        - ++ nextEpochPoolCount (contracts/Staking.sol#223)
        - _moveMemberBalance(mbr_,MoveBalanceDirection.INTO)
(contracts/Staking.sol#220)
                -
unclaimedRewards[poolObj.beneficiary][RewardSrc.POOL_OWNERSHIP] += toCollect
(contracts/rewards/MemberRewardsAdapter.sol#218)
                - unclaimedRewards[member][RewardSrc.REGULAR_DELEGATOR] +=
toCollect (contracts/rewards/MemberRewardsAdapter.sol#220)
Reentrancy in Staking.delegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/Staking.sol#342-353):
        External calls:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#344)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - token.safeTransferFrom(from,to,value)
(contracts/deposits/ERC20Utils.sol#20)
```

```
                - ERC20Utils.strictTransferFrom(erc20,account,address(this),amount) (contracts/deposits/TransfersAdapter.sol#68)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#344)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
        State variables written after the call(s):
        - _delegate(mbr_) (contracts/Staking.sol#345)
                - delegationObj.currencies[ca_.erc20] = changeCurrencyFn(delegationObj.currencies[ca_.erc20],ca_.amount) (contracts/deposits/DelegationsAdapter.sol#52)
        - _delegate(mbr_) (contracts/Staking.sol#345)
                - unclaimedRewards[poolObj.beneficiary][RewardSrc.POOL_OWNERSHIP] += toCollect (contracts/rewards/MemberRewardsAdapter.sol#218)
                - unclaimedRewards[member][RewardSrc.REGULAR_DELEGATOR] += toCollect (contracts/rewards/MemberRewardsAdapter.sol#220)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Staking.claim(bytes32[],Finalizer.RewardSrc) (contracts/Staking.sol#525-536):
        External calls:
        - _strictTransferTo(_msgSender(),CurrencyAmount(rewardsToken,unclaimed)) (contracts/Staking.sol#533)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - ca_.erc20.safeTransfer(account,ca_.amount) (contracts/deposits/TransfersAdapter.sol#52)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
        External calls sending eth:
        - _strictTransferTo(_msgSender(),CurrencyAmount(rewardsToken,unclaimed)) (contracts/Staking.sol#533)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
        Event emitted after the call(s):
        - Claim(_msgSender(),source,unclaimed) (contracts/Staking.sol#535)
Reentrancy in Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16) (contracts/Staking.sol#209-230):
```

```
        External calls:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#219)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - token.safeTransferFrom(from,to,value)
(contracts/deposits/ERC20Utils.sol#20)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/deposits/TransfersAdapter.sol#68)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#219)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        -
CollectMemberRewards(member,poolId,toCollect,memberDeposit.collectedEpoch)
(contracts/rewards/MemberRewardsAdapter.sol#228)
                - _moveMemberBalance(mbr_,MoveBalanceDirection.INTO)
(contracts/Staking.sol#220)
        - CreateNewPool(mbr_.member,mbr_.poolId,mbr_.currencies)
(contracts/Staking.sol#225)
        - SetPoolBeneficiary(poolId,beneficiary) (contracts/Staking.sol#298)
                - _setPoolBeneficiary(mbr_.poolId,beneficiary)
(contracts/Staking.sol#228)
        - SetPoolCommissionFee(poolId,commission) (contracts/Staking.sol#271)
                - _setPoolCommission(mbr_.poolId,commissionFee)
(contracts/Staking.sol#229)
Reentrancy in Staking.delegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/Staking.sol#342-353):
        External calls:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#344)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - token.safeTransferFrom(from,to,value)
(contracts/deposits/ERC20Utils.sol#20)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/deposits/TransfersAdapter.sol#68)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/Staking.sol#344)
```

```
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        -
CollectMemberRewards(member,poolId,toCollect,memberDeposit.collectedEpoch)
(contracts/rewards/MemberRewardsAdapter.sol#228)
                - _delegate(mbr_) (contracts/Staking.sol#345)
        - Delegate(mbr_.member,mbr_.poolId,mbr_.currencies)
(contracts/Staking.sol#352)
Reentrancy in Staking.depositFunds(uint256) (contracts/Staking.sol#146-149):
        External calls:
        - depositedRewardFunds += amount =
_strictTransferFrom(_msgSender(),rewardsToken,amount)
(contracts/Staking.sol#147)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - token.safeTransferFrom(from,to,value)
(contracts/deposits/ERC20Utils.sol#20)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/deposits/TransfersAdapter.sol#68)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - depositedRewardFunds += amount =
_strictTransferFrom(_msgSender(),rewardsToken,amount)
(contracts/Staking.sol#147)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        - DepositFunds(amount) (contracts/Staking.sol#148)
Reentrancy in Staking.executeWithdrawRequest(bytes32)
(contracts/Staking.sol#501-516):
        External calls:
        - _strictTransferTo(_msgSender(),currencies[it])
(contracts/Staking.sol#511)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - ca_.erc20.safeTransfer(account,ca_.amount)
(contracts/deposits/TransfersAdapter.sol#52)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferTo(_msgSender(),currencies[it])
(contracts/Staking.sol#511)
```

```
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        - WithdrawRequestExecuted(_msgSender(),poolId)
(contracts/Staking.sol#515)
Reentrancy in Staking.unjail(bytes32) (contracts/Staking.sol#431-439):
        External calls:
        - depositFunds(getConfig(POOL_UNJAIL_FEE))
(contracts/Staking.sol#435)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - token.safeTransferFrom(from,to,value)
(contracts/deposits/ERC20Utils.sol#20)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/deposits/TransfersAdapter.sol#68)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - depositFunds(getConfig(POOL_UNJAIL_FEE))
(contracts/Staking.sol#435)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        - UnjailPool(poolId) (contracts/Staking.sol#438)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vuln
erabilities-3

Staking.filterPoolIdsIdle(bytes32[]) (contracts/Staking.sol#155-167) uses
assembly
        - INLINE ASM (contracts/Staking.sol#164-166)
AddressUpgradeable._revert(bytes,string)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#206-218) uses assembly
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#211-214)
StringsUpgradeable.toString(uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.so
l#18-38) uses assembly
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.so
l#24-26)
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.so
l#30-32)
```

```
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) uses assembly
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#66-70)
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#86-93)
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#100-109)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/rewards/Finalizer.sol#209-258) has costly operations inside a
loop:
        - depositedRewardFunds -= chainReward
(contracts/rewards/Finalizer.sol#241)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operatio
ns-inside-a-loop

Low level call in AddressUpgradeable.sendValue(address,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#60-65):
        - (success) = recipient.call{value: amount}()
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#63)
Low level call in
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#128-137):
        - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#155-162):
        - (success,returndata) = target.staticcall(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#160)
```

Revision Audit

Timeline: 2 Feb 2023

Methods: Manual & Automated Review

Total Issues:

Critical/High: 0

Medium: 1

Low: 0

Gas Optimizations: 0

Informational: 2

Staking

1. `function undelegate(MoveBalanceRequest memory mbr_)`: Frontrun attack, pool owner can monitor the mempool and when a jail transaction is pending, he can execute `undelegate` and retrieve all funds after the execution time.

```
function undelegate(MoveBalanceRequest memory mbr_)
        external
        override
        onlyRole(NODE_MANAGER_ROLE)
        payPoolOpFee(mbr_.poolId)
        returns (bool poolDeleted)
    {
        poolDeleted = _isPoolOwner(mbr_.member, mbr_.poolId);
        // cannot undelegate as owner if pool is jailed
        require(!poolDeleted || !isPoolJailed(mbr_.poolId)), "BS107");

        // ensure `_undelegate` not disabled by short-circuit evaluation
        poolDeleted = (_undelegate(mbr_) == 0) && poolDeleted;
        // pledged balance != 0 acts as a proxy for pool liveness
        if (poolDeleted) {
            // pool does not require finalization from next epoch onwards
            --_nextEpochPoolCount;
            // remember last epoch of cumulative rewards that will ever be
set
            _lastStoredCRwd[mbr_.poolId] = currentEpoch() + 1;
        }

_createWithdrawRequest(_poolsArchive[mbr_.poolId].members[mbr_.member].withdr
awRequest, mbr_);

        emit Undelegate(mbr_.member, mbr_.poolId, mbr_.currencies);
    }
```

Recommendation: Softbinator recommends adding a check within the `executeWithdrawRequest(bytes32 poolId)` function, to revert in case the poolId is jailed. If this is implemented, then the pool owner can undelegate his funds, but he will not get them until he pays the jail fee.

Resolution: Resolved

2. `function executeWithdrawRequest(bytes32 poolId)`: CEI pattern not respected

```
function executeWithdrawRequest(bytes32 poolId) external payPoolOpFee(poolId)
{
        WithdrawRequest storage withdrawObj =
_poolsArchive[poolId].members[_msgSender()].withdrawRequest;
        require(withdrawObj.currencies.length > 0, "BS115");
        require(withdrawObj.executeTime <= block.timestamp, "BS116");

        CurrencyAmount[] memory currencies = withdrawObj.currencies;
        for (uint256 it; it < currencies.length; ++it) {
            _strictTransferTo(_msgSender(), currencies[it]);
        }
        delete withdrawObj.currencies;

        emit WithdrawRequestExecuted(_msgSender(), poolId);
    }
```

Recommendation: Softbinator recommends to move **delete withdrawObj.currencies;** above the transfer call:

```
CurrencyAmount[] memory currencies = withdrawObj.currencies;
delete withdrawObj.currencies;
for (uint256 it; it < currencies.length; ++it) {
    _strictTransferTo(_msgSender(), currencies[it]);
}
```

Resolution: Resolved

3. `function createNewPool(MoveBalanceRequest memory mbr_,address beneficiary,uint16 commissionFee)`: CEI pattern not respected

```
function createNewPool(
        MoveBalanceRequest memory mbr_,
        address beneficiary,
        uint16 commissionFee
```

```
    ) external override onlyRole(NODE_MANAGER_ROLE) payPoolOpFee(mbr_.poolId)
{
        // `owner` acts as a proxy for pool creation
        require(_isPoolOwner(address(0), mbr_.poolId), "BS100");
        // assign owner before bonding pledge to pool
        _poolsArchive[mbr_.poolId].meta.owner = mbr_.member;

        _strictTransferFrom(mbr_);
        _moveMemberBalance(mbr_, MoveBalanceDirection.INTO);

        // pool requires finalization from next epoch onwards
        ++_nextEpochPoolCount;

        emit CreateNewPool(mbr_.member, mbr_.poolId, mbr_.currencies);

        // emit params config events after the pool creation one
        _setPoolBeneficiary(mbr_.poolId, beneficiary);
        _setPoolCommission(mbr_.poolId, commissionFee);
    }
```

Recommendation: Softbinator recommends to move external calls at the end:

```
.....
_strictTransferFrom(mbr_);
_moveMemberBalance(mbr_, MoveBalanceDirection.INTO);
}
```

Resolution: Resolved

Automated Review - Slither

We exclude a few detectors which weren't relevant for the contracts:

naming-convention,solc-version, pragma, timestamp, uninitialized-local

Below is the generated raport, from which we excluded the analysis of Openzepellin contracts regarding dead-code.

Excepting the issues presented above, we don't believe that the results found by slither can represent a security risk for this contract, but they can be addressed by the developing team

```
$ slither ./contracts/staking/Staking.sol --solc-remaps
'@openzeppelin=node_modules/@openzeppelin hardhat=node_modules/hardhat'
--exclude naming-convention,solc-version,pragma,timestamp,uninitialized-local
Compilation warnings/errors on ./contracts/staking/Staking.sol:
```

```
Warning: Contract code size is 45217 bytes and exceeds 24576 bytes (a limit
introduced in Spurious Dragon). This contract may not be deployable on
Mainnet. Consider enabling the optimizer (with a low "runs" value!), turning
off revert strings, or using libraries.
 --> contracts/staking/Staking.sol:7:1:
  |
7 | contract Staking is IPoolCreator, DelegationsAdapter {
  | ^ (Relevant source part starts here and spans across multiple lines).



ERC20Utils.strictTransferFrom(IERC20Upgradeable,address,address,uint256)
(contracts/libraries/ERC20Utils.sol#13-22) uses arbitrary from in
transferFrom: token.safeTransferFrom(from,to,value)
(contracts/libraries/ERC20Utils.sol#20)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-
in-transferfrom

Reentrancy in
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/staking/Staking.sol#234-255):
        External calls:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#244)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/staking/deposits/TransfersAdapter.sol#72)
                - token.safeTransferFrom(from,to,value)
(contracts/libraries/ERC20Utils.sol#20)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#244)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        State variables written after the call(s):
        - _moveMemberBalance(mbr_,MoveBalanceDirection.INTO)
(contracts/staking/Staking.sol#245)
                - delegationObj.currencies[ca_.erc20] =
changeCurrencyFn(delegationObj.currencies[ca_.erc20],ca_.amount)
(contracts/staking/deposits/DelegationsAdapter.sol#51)
        StakingStorage._poolsArchive
(contracts/staking/StakingStorage.sol#61) can be used in cross function
reentrancies:
        - Staking._cancelWithdrawRequest(address,bytes32,bytes32)
(contracts/staking/Staking.sol#487-499)
```

```
        -
DelegationsAdapter._changeCurrencyAmount(IMoveBalanceStructs.MoveBalanceReque
st,function(uint256,uint256) returns(uint256))
(contracts/staking/deposits/DelegationsAdapter.sol#35-53)
        -
DelegationsAdapter._changeMemberBalance(IMoveBalanceStructs.MoveBalanceReques
t,function(DepositsAdapter.DeferredDeposit,uint256) returns(uint256))
(contracts/staking/deposits/DelegationsAdapter.sol#60-86)
        - Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#250-305)
        - Staking._isPoolOwner(address,bytes32)
(contracts/staking/Staking.sol#268-270)
        - Staking._setPoolBeneficiary(bytes32,address)
(contracts/staking/Staking.sol#319-324)
        - Staking._setPoolCommission(bytes32,uint16)
(contracts/staking/Staking.sol#291-297)
        - MemberRewardsAdapter._syncMemberRewards(address,bytes32)
(contracts/staking/rewards/MemberRewardsAdapter.sol#226-243)
        - MemberRewardsAdapter.computeMemberReward(address,bytes32[])
(contracts/staking/rewards/MemberRewardsAdapter.sol#158-166)
        -
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/staking/Staking.sol#234-255)
        - Staking.executeWithdrawRequest(bytes32)
(contracts/staking/Staking.sol#527-539)
        - Staking.getMemberBalance(address,bytes32)
(contracts/staking/Staking.sol#199-201)
        - Staking.getMemberCurrency(address,bytes32,IERC20Upgradeable)
(contracts/staking/Staking.sol#209-215)
        - Staking.getMemberWithdrawRequest(address,bytes32)
(contracts/staking/Staking.sol#222-224)
        - Staking.isPoolJailed(bytes32)
(contracts/staking/Staking.sol#339-341)
        - Staking.isPoolOperative(bytes32)
(contracts/staking/Staking.sol#331-333)
        - Staking.jail(bytes32) (contracts/staking/Staking.sol#444-449)
        - StakingStorage.poolsArchive(bytes32)
(contracts/staking/StakingStorage.sol#69-71)
        - Staking.setPoolBeneficiary(bytes32,address)
(contracts/staking/Staking.sol#304-312)
        - Staking.setPoolCommission(bytes32,uint16)
(contracts/staking/Staking.sol#278-284)
        -
MemberRewardsAdapter.syncMemberRewards(bytes32[],address,Finalizer.RewardSrc)
(contracts/staking/rewards/MemberRewardsAdapter.sol#252-264)
        - Staking.undelegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/staking/Staking.sol#396-419)
        - Staking.unjail(bytes32) (contracts/staking/Staking.sol#456-465)
        - _setPoolBeneficiary(mbr_.poolId,beneficiary)
(contracts/staking/Staking.sol#253)
                - _poolsArchive[poolId].meta.beneficiary = beneficiary
(contracts/staking/Staking.sol#321)
```

```
        StakingStorage._poolsArchive
(contracts/staking/StakingStorage.sol#61) can be used in cross function
reentrancies:
        - Staking._cancelWithdrawRequest(address,bytes32,bytes32)
(contracts/staking/Staking.sol#487-499)
        -
DelegationsAdapter._changeCurrencyAmount(IMoveBalanceStructs.MoveBalanceReque
st,function(uint256,uint256) returns(uint256))
(contracts/staking/deposits/DelegationsAdapter.sol#35-53)
        -
DelegationsAdapter._changeMemberBalance(IMoveBalanceStructs.MoveBalanceReques
t,function(DepositsAdapter.DeferredDeposit,uint256) returns(uint256))
(contracts/staking/deposits/DelegationsAdapter.sol#60-86)
        - Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#250-305)
        - Staking._isPoolOwner(address,bytes32)
(contracts/staking/Staking.sol#268-270)
        - Staking._setPoolBeneficiary(bytes32,address)
(contracts/staking/Staking.sol#319-324)
        - Staking._setPoolCommission(bytes32,uint16)
(contracts/staking/Staking.sol#291-297)
        - MemberRewardsAdapter._syncMemberRewards(address,bytes32)
(contracts/staking/rewards/MemberRewardsAdapter.sol#226-243)
        - MemberRewardsAdapter.computeMemberReward(address,bytes32[])
(contracts/staking/rewards/MemberRewardsAdapter.sol#158-166)
        -
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/staking/Staking.sol#234-255)
        - Staking.executeWithdrawRequest(bytes32)
(contracts/staking/Staking.sol#527-539)
        - Staking.getMemberBalance(address,bytes32)
(contracts/staking/Staking.sol#199-201)
        - Staking.getMemberCurrency(address,bytes32,IERC20Upgradeable)
(contracts/staking/Staking.sol#209-215)
        - Staking.getMemberWithdrawRequest(address,bytes32)
(contracts/staking/Staking.sol#222-224)
        - Staking.isPoolJailed(bytes32)
(contracts/staking/Staking.sol#339-341)
        - Staking.isPoolOperative(bytes32)
(contracts/staking/Staking.sol#331-333)
        - Staking.jail(bytes32) (contracts/staking/Staking.sol#444-449)
        - StakingStorage.poolsArchive(bytes32)
(contracts/staking/StakingStorage.sol#69-71)
        - Staking.setPoolBeneficiary(bytes32,address)
(contracts/staking/Staking.sol#304-312)
        - Staking.setPoolCommission(bytes32,uint16)
(contracts/staking/Staking.sol#278-284)
        -
MemberRewardsAdapter.syncMemberRewards(bytes32[],address,Finalizer.RewardSrc)
(contracts/staking/rewards/MemberRewardsAdapter.sol#252-264)
        - Staking.undelegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/staking/Staking.sol#396-419)
        - Staking.unjail(bytes32) (contracts/staking/Staking.sol#456-465)
```

```
        - _setPoolCommission(mbr_.poolId,commissionFee)
(contracts/staking/Staking.sol#254)
                - _poolsArchive[poolId].meta.commissionFee = commission
(contracts/staking/Staking.sol#293)
                - _poolsArchive[poolId].meta.commissionFeeEpochSet =
currentEpoch() (contracts/staking/Staking.sol#294)
        StakingStorage._poolsArchive
(contracts/staking/StakingStorage.sol#61) can be used in cross function
reentrancies:
        - Staking._cancelWithdrawRequest(address,bytes32,bytes32)
(contracts/staking/Staking.sol#487-499)
        -
DelegationsAdapter._changeCurrencyAmount(IMoveBalanceStructs.MoveBalanceReque
st,function(uint256,uint256) returns(uint256))
(contracts/staking/deposits/DelegationsAdapter.sol#35-53)
        -
DelegationsAdapter._changeMemberBalance(IMoveBalanceStructs.MoveBalanceReques
t,function(DepositsAdapter.DeferredDeposit,uint256) returns(uint256))
(contracts/staking/deposits/DelegationsAdapter.sol#60-86)
        - Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#250-305)
        - Staking._isPoolOwner(address,bytes32)
(contracts/staking/Staking.sol#268-270)
        - Staking._setPoolBeneficiary(bytes32,address)
(contracts/staking/Staking.sol#319-324)
        - Staking._setPoolCommission(bytes32,uint16)
(contracts/staking/Staking.sol#291-297)
        - MemberRewardsAdapter._syncMemberRewards(address,bytes32)
(contracts/staking/rewards/MemberRewardsAdapter.sol#226-243)
        - MemberRewardsAdapter.computeMemberReward(address,bytes32[])
(contracts/staking/rewards/MemberRewardsAdapter.sol#158-166)
        -
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/staking/Staking.sol#234-255)
        - Staking.executeWithdrawRequest(bytes32)
(contracts/staking/Staking.sol#527-539)
        - Staking.getMemberBalance(address,bytes32)
(contracts/staking/Staking.sol#199-201)
        - Staking.getMemberCurrency(address,bytes32,IERC20Upgradeable)
(contracts/staking/Staking.sol#209-215)
        - Staking.getMemberWithdrawRequest(address,bytes32)
(contracts/staking/Staking.sol#222-224)
        - Staking.isPoolJailed(bytes32)
(contracts/staking/Staking.sol#339-341)
        - Staking.isPoolOperative(bytes32)
(contracts/staking/Staking.sol#331-333)
        - Staking.jail(bytes32) (contracts/staking/Staking.sol#444-449)
        - StakingStorage.poolsArchive(bytes32)
(contracts/staking/StakingStorage.sol#69-71)
        - Staking.setPoolBeneficiary(bytes32,address)
(contracts/staking/Staking.sol#304-312)
        - Staking.setPoolCommission(bytes32,uint16)
(contracts/staking/Staking.sol#278-284)
```

```
        -
MemberRewardsAdapter.syncMemberRewards(bytes32[],address,Finalizer.RewardSrc)
(contracts/staking/rewards/MemberRewardsAdapter.sol#252-264)
        - Staking.undelegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/staking/Staking.sol#396-419)
        - Staking.unjail(bytes32) (contracts/staking/Staking.sol#456-465)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vuln
erabilities

Finalizer._computeEpochRewards(Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#176-180) performs a multiplication
on the result of a division:
        - epochRewards = (snapshot.depositedTotal *
getConfig(AVERAGE_TOKEN_APY)) / MAX_PPM
(contracts/staking/rewards/Finalizer.sol#177)
        - epochRewards = (epochRewards * snapshot.prevEpochLength) /
ONE_YEAR_SEC (contracts/staking/rewards/Finalizer.sol#178)
Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#250-305) performs a multiplication
on the result of a division:
        - membersReward = (membersReward * performance) / MAX_PPM
(contracts/staking/rewards/Finalizer.sol#283)
        - ownerReward = (membersReward * poolObj.commissionFee) / MAX_PPM
(contracts/staking/rewards/Finalizer.sol#291)
Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#250-305) performs a multiplication
on the result of a division:
        - membersReward = (membersReward * performance) / MAX_PPM
(contracts/staking/rewards/Finalizer.sol#283)
        - cr_.rewardRatio =
_poolCumulativeReward[poolId][snapshot.finalizerEpoch - 1].rewardRatio +
SafeCastUpgradeable.toUint128((membersReward * ONE_TOKEN_UNITS) / delegated)
(contracts/staking/rewards/Finalizer.sol#299-301)
Finalizer._computePoolEpochRewards(uint256,uint256,Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#314-330) performs a multiplication
on the result of a division:
        - poolRewards = (pledged * (delegated - poolRewards)) /
snapshot.saturationCap (contracts/staking/rewards/Finalizer.sol#323)
        - poolRewards = (poolRewards * snapshot.pledgeFactorNum) /
snapshot.pledgeFactorDenom (contracts/staking/rewards/Finalizer.sol#324)
Finalizer._computePoolEpochRewards(uint256,uint256,Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#314-330) performs a multiplication
on the result of a division:
        - poolRewards = (poolRewards * snapshot.pledgeFactorNum) /
snapshot.pledgeFactorDenom (contracts/staking/rewards/Finalizer.sol#324)
        - poolRewards = (poolRewards * (snapshot.pledgeFactorDenom)) /
(snapshot.pledgeFactorDenom + snapshot.pledgeFactorNum)
(contracts/staking/rewards/Finalizer.sol#326-328)
Finalizer._computePoolEpochRewards(uint256,uint256,Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#314-330) performs a multiplication
on the result of a division:
```

```
        - poolRewards = (poolRewards * (snapshot.pledgeFactorDenom)) /
(snapshot.pledgeFactorDenom + snapshot.pledgeFactorNum)
(contracts/staking/rewards/Finalizer.sol#326-328)
        - poolRewards = (snapshot.prevEpochRewards * poolRewards) /
snapshot.depositedTotal (contracts/staking/rewards/Finalizer.sol#329)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse = (3 * denominator) ^ 2
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
```

```
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - denominator = denominator / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#102)
        - inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) performs a multiplication on the result of a division:
        - prod0 = prod0 / twos
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#105)
        - result = prod0 * inverse
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#132)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-m
ultiply

MemberRewardsAdapter._computeMemberRewardOverInterval(bytes32,uint256,uint64,
uint64).endEpoch (contracts/staking/rewards/MemberRewardsAdapter.sol#137)
shadows:
        - Finalizer.endEpoch()
(contracts/staking/rewards/Finalizer.sol#186-215) (function)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing

AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#128-137) has external calls inside a loop: (success,returndata) =
target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a
-loop

Reentrancy in
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/staking/Staking.sol#234-255):
        External calls:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#244)
```

```
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/staking/deposits/TransfersAdapter.sol#72)
                - token.safeTransferFrom(from,to,value)
(contracts/libraries/ERC20Utils.sol#20)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#244)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        State variables written after the call(s):
        - ++ _nextEpochPoolCount (contracts/staking/Staking.sol#248)
        - _moveMemberBalance(mbr_,MoveBalanceDirection.INTO)
(contracts/staking/Staking.sol#245)
                -
_unclaimedRewards[poolObj.meta.beneficiary][RewardSrc.POOL_OWNERSHIP] +=
toCollect (contracts/staking/rewards/MemberRewardsAdapter.sol#232)
                - _unclaimedRewards[member][RewardSrc.REGULAR_DELEGATOR] +=
toCollect (contracts/staking/rewards/MemberRewardsAdapter.sol#234)
Reentrancy in Staking.delegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/staking/Staking.sol#367-378):
        External calls:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#369)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/staking/deposits/TransfersAdapter.sol#72)
                - token.safeTransferFrom(from,to,value)
(contracts/libraries/ERC20Utils.sol#20)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#369)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        State variables written after the call(s):
        - _delegate(mbr_) (contracts/staking/Staking.sol#370)
                - delegationObj.currencies[ca_.erc20] =
changeCurrencyFn(delegationObj.currencies[ca_.erc20],ca_.amount)
(contracts/staking/deposits/DelegationsAdapter.sol#51)
        - _delegate(mbr_) (contracts/staking/Staking.sol#370)
```

```
                -
_unclaimedRewards[poolObj.meta.beneficiary][RewardSrc.POOL_OWNERSHIP] +=
toCollect (contracts/staking/rewards/MemberRewardsAdapter.sol#232)
                - _unclaimedRewards[member][RewardSrc.REGULAR_DELEGATOR] +=
toCollect (contracts/staking/rewards/MemberRewardsAdapter.sol#234)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vuln
erabilities-2

Reentrancy in Staking.claim(bytes32[],Finalizer.RewardSrc)
(contracts/staking/Staking.sol#548-559):
        External calls:
        -
_strictTransferTo(_msgSender(),CurrencyAmount(_rewardsToken,unclaimed))
(contracts/staking/Staking.sol#556)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - ca_.erc20.safeTransfer(account,ca_.amount)
(contracts/staking/deposits/TransfersAdapter.sol#56)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        -
_strictTransferTo(_msgSender(),CurrencyAmount(_rewardsToken,unclaimed))
(contracts/staking/Staking.sol#556)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        - Claim(_msgSender(),source,unclaimed)
(contracts/staking/Staking.sol#558)
Reentrancy in
Staking.createNewPool(IMoveBalanceStructs.MoveBalanceRequest,address,uint16)
(contracts/staking/Staking.sol#234-255):
        External calls:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#244)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/staking/deposits/TransfersAdapter.sol#72)
                - token.safeTransferFrom(from,to,value)
(contracts/libraries/ERC20Utils.sol#20)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#244)
```

```
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        -
CollectMemberRewards(member,poolId,toCollect,memberDeposit.collectedEpoch)
(contracts/staking/rewards/MemberRewardsAdapter.sol#242)
                - _moveMemberBalance(mbr_,MoveBalanceDirection.INTO)
(contracts/staking/Staking.sol#245)
        - CreateNewPool(mbr_.member,mbr_.poolId,mbr_.currencies)
(contracts/staking/Staking.sol#250)
        - SetPoolBeneficiary(poolId,beneficiary)
(contracts/staking/Staking.sol#323)
                - _setPoolBeneficiary(mbr_.poolId,beneficiary)
(contracts/staking/Staking.sol#253)
        - SetPoolCommissionFee(poolId,commission)
(contracts/staking/Staking.sol#296)
                - _setPoolCommission(mbr_.poolId,commissionFee)
(contracts/staking/Staking.sol#254)
Reentrancy in Staking.delegate(IMoveBalanceStructs.MoveBalanceRequest)
(contracts/staking/Staking.sol#367-378):
        External calls:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#369)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/staking/deposits/TransfersAdapter.sol#72)
                - token.safeTransferFrom(from,to,value)
(contracts/libraries/ERC20Utils.sol#20)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferFrom(mbr_) (contracts/staking/Staking.sol#369)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        -
CollectMemberRewards(member,poolId,toCollect,memberDeposit.collectedEpoch)
(contracts/staking/rewards/MemberRewardsAdapter.sol#242)
                - _delegate(mbr_) (contracts/staking/Staking.sol#370)
        - Delegate(mbr_.member,mbr_.poolId,mbr_.currencies)
(contracts/staking/Staking.sol#377)
Reentrancy in Staking.depositFunds(uint256)
(contracts/staking/Staking.sol#171-174):
        External calls:
        - _depositedRewardFunds += amount =
_strictTransferFrom(_msgSender(),_rewardsToken,amount)
(contracts/staking/Staking.sol#172)
```

```
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/staking/deposits/TransfersAdapter.sol#72)
                - token.safeTransferFrom(from,to,value)
(contracts/libraries/ERC20Utils.sol#20)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _depositedRewardFunds += amount =
_strictTransferFrom(_msgSender(),_rewardsToken,amount)
(contracts/staking/Staking.sol#172)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        - DepositFunds(amount) (contracts/staking/Staking.sol#173)
Reentrancy in Staking.executeWithdrawRequest(bytes32)
(contracts/staking/Staking.sol#527-539):
        External calls:
        - _strictTransferTo(_msgSender(),currencies[it])
(contracts/staking/Staking.sol#534)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
                - ca_.erc20.safeTransfer(account,ca_.amount)
(contracts/staking/deposits/TransfersAdapter.sol#56)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - _strictTransferTo(_msgSender(),currencies[it])
(contracts/staking/Staking.sol#534)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        - WithdrawRequestExecuted(_msgSender(),poolId)
(contracts/staking/Staking.sol#538)
Reentrancy in Staking.unjail(bytes32)
(contracts/staking/Staking.sol#456-465):
        External calls:
        - depositFunds(getConfig(POOL_UNJAIL_FEE))
(contracts/staking/Staking.sol#461)
                - returndata = address(token).functionCall(data,SafeERC20:
low-level call failed)
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20
Upgradeable.sol#110)
```

```
                -
ERC20Utils.strictTransferFrom(erc20,account,address(this),amount)
(contracts/staking/deposits/TransfersAdapter.sol#72)
                - token.safeTransferFrom(from,to,value)
(contracts/libraries/ERC20Utils.sol#20)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        External calls sending eth:
        - depositFunds(getConfig(POOL_UNJAIL_FEE))
(contracts/staking/Staking.sol#461)
                - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
        Event emitted after the call(s):
        - UnjailPool(poolId) (contracts/staking/Staking.sol#464)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vuln
erabilities-3

Staking.filterPoolIdsIdle(bytes32[]) (contracts/staking/Staking.sol#180-192)
uses assembly
        - INLINE ASM (contracts/staking/Staking.sol#189-191)
AddressUpgradeable._revert(bytes,string)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#206-218) uses assembly
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#211-214)
StringsUpgradeable.toString(uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.so
l#18-38) uses assembly
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.so
l#24-26)
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.so
l#30-32)
MathUpgradeable.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#55-135) uses assembly
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#66-70)
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#86-93)
        - INLINE ASM
(node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.
sol#100-109)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Finalizer._finalizePool(bytes32,uint16,Finalizer.EpochSnapshot)
(contracts/staking/rewards/Finalizer.sol#250-305) has costly operations
inside a loop:
        - _depositedRewardFunds -= chainReward
(contracts/staking/rewards/Finalizer.sol#288)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operatio
ns-inside-a-loop


Low level call in AddressUpgradeable.sendValue(address,uint256)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#60-65):
        - (success) = recipient.call{value: amount}()
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#63)
Low level call in
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#128-137):
        - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#135)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#155-162):
        - (success,returndata) = target.staticcall(data)
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.so
l#160)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable Staking._rewardsToken (contracts/staking/Staking.sol#11) is too
similar to
Staking.initialize(address,IERC20Upgradeable,uint256).rewardsToken_
(contracts/staking/Staking.sol#100)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
too-similar
```