OZYS
KLAYswap Lending
Security Analysis Report

Prepared by

78ResearchLab

RedSpider

June 03, 2024

TABLE OF CONTENTS

PROJECT OVERALL

About Project

This project is a lending service based on AAVE V3, a decentralized non-custodial liquidity protocol where users can participate as suppliers, borrowers, or liquidators. Ozys added a custom reward token manager and reward transfer strategy to distribute the KSP token as a reward.

Target Summary

Name	KLAYswap Lending
Website	https://ozys.io/
Repository	https://git.ozys.work/silicon/swap/new-lending-contract/-/tree/dev
Commit	d4b4d4406ab4f8cdcd1a20bcb1a06d1d4382ea86
Network	Klaytn
Languages	Solidity
Method	Source code auditing
Timeline	May 13, 2024 ~ May 17, 2024

SCOPE

We were provided with the GitHub repository to review. Although the base commit for the audit was set to d4b4d440, we also reviewed updates and fixes that were committed subsequently. This project is based on AAVE V3, which has been extensively audited. Therefore, we focused on the new features written by Ozys, including the integration of reward transfer strategy and reward token manager.

Source code

Name	commit
KLAYswap Lending	d4b4d4406ab4f8cdcd1a20bcb1a06d1d4382ea86

```
./contracts/
 └──── periphery
       └──── rewards
             ├──── LendingRewardTokenManager.sol
             └──── transfer-strategies
                   └──── PullRewardsTransferStrategyForSwap.sol
```

RISK CLASSIFICATION

Severity

Our risk classification is based on Severity Categorization of code4ena.

High ●

Assets can be stolen, lost, compromised directly or indirectly via a valid attack path (e.g. Malicious Input Handling, Escalation of privileges, Arithmetic).

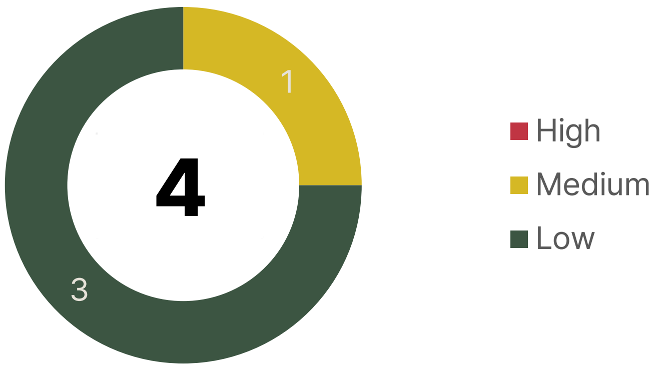
Medium ●

Assets not at direct risk, but the function of the protocol or its availability could be impacted, or leak value with a hypothetical attack path with stated assumptions, but external requirements.

Low ●

Assets are not at risk. User mistake, misuse of privileges, governance risk fall under this grade.

FINDINGS BREAKDOWN



Severity	Acknowledged	Fixed	Total
● High	0	0	0
● Medium	1	0	1
● Low	0	3	3
			4

* Fixed: Risk is fixed by Ozys.

* Acknowledged: Ozys has recognized the risk but has not addressed it, as it poses only a minor impact.

FINDINGS

● MEDIUM

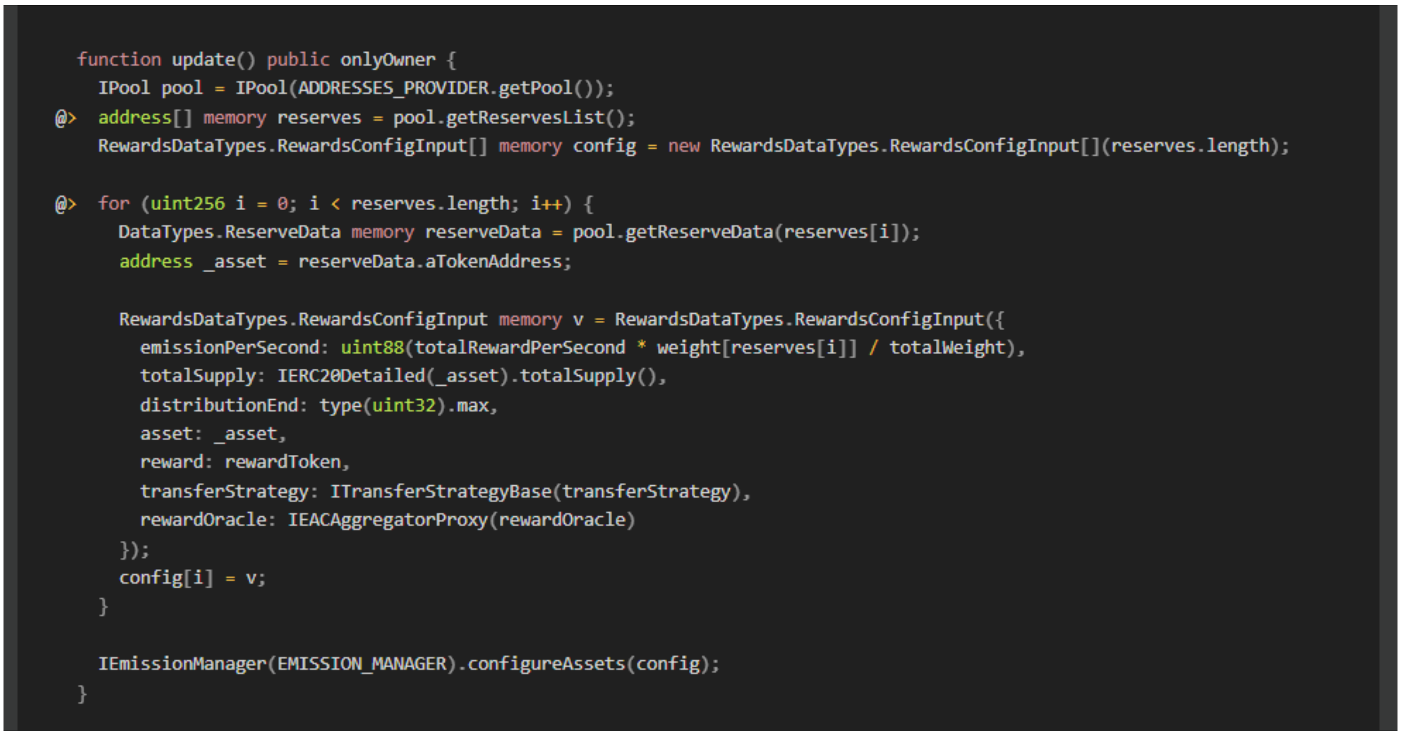
M-01. LendingRewardTokenManager: Cannot delete the reward setting of a deleted pool Acknowledged

IMPACT

Cannot delete the reward settings for a deleted pool. AAVE V3 allows rewards to continue to be generated/claimed regardless of whether the pool is deleted/inactive/paused, so if the reward settings are not deleted, rewards will continue to be generated/claimed.

DESCRIPTION

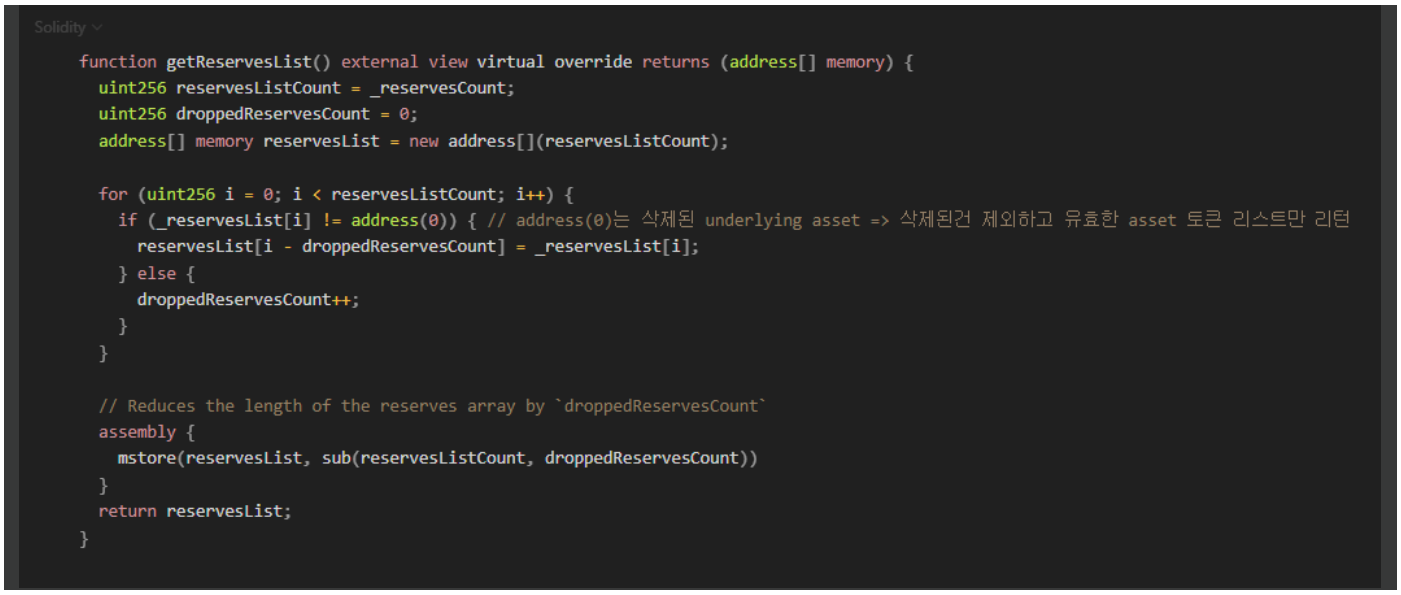
When changing the reward settings, all currently registered underlying asset tokens are retrieved using the `pool.getReservesList` function. Then, the reward setting for each underlying asset is set.



```
  function update() public onlyOwner {
    IPool pool = IPool(ADDRESSES_PROVIDER.getPool());
@>  address[] memory reserves = pool.getReservesList();
    RewardsDataTypes.RewardsConfigInput[] memory config = new RewardsDataTypes.RewardsConfigInput[](reserves.length);

@>  for (uint256 i = 0; i < reserves.length; i++) {
      DataTypes.ReserveData memory reserveData = pool.getReserveData(reserves[i]);
      address _asset = reserveData.aTokenAddress;

      RewardsDataTypes.RewardsConfigInput memory v = RewardsDataTypes.RewardsConfigInput({
        emissionPerSecond: uint88(totalRewardPerSecond * weight[reserves[i]] / totalWeight),
        totalSupply: IERC20Detailed(_asset).totalSupply(),
        distributionEnd: type(uint32).max,
        asset: _asset,
        reward: rewardToken,
        transferStrategy: ITransferStrategyBase(transferStrategy),
        rewardOracle: IEACAggregatorProxy(rewardOracle)
      });
      config[i] = v;
    }

    IEmissionManager(EMISSION_MANAGER).configureAssets(config);
  }
```

File 1 - periphery/rewards/LendingRewardTokenManager.sol#L91-110 Function: update

The `Pool.getReservesList` function includes only underlying asset tokens that have not been deleted (including paused and inactive states), and does not include underlying assets that have been registered and then deleted.


Solidity

```
function getReservesList() external view virtual override returns (address[] memory) {
  uint256 reservesListCount = _reservesCount;
  uint256 droppedReservesCount = 0;
  address[] memory reservesList = new address[](reservesListCount);

  for (uint256 i = 0; i < reservesListCount; i++) {
    if (_reservesList[i] != address(0)) { // address(0)는 삭제된 underlying asset => 삭제된건 제외하고 유효한 asset 토큰 리스트만 리턴
      reservesList[i - droppedReservesCount] = _reservesList[i];
    } else {
      droppedReservesCount++;
    }
  }

  // Reduces the length of the reserves array by `droppedReservesCount`
  assembly {
    mstore(reservesList, sub(reservesListCount, droppedReservesCount))
  }
  return reservesList;
}
```

File 2 - core/protocol/pool/Pool.sol Function: getReservesList

Therefore, if you delete an underlying asset that had previously set reward information using the `Pool.dropReserve` function and then try to set the weight of the deleted pool to 0 through `LendingRewardTokenManager.manage` and `LendingRewardTokenManager.update`, it will fail. This is because the return value of the `Pool.getReservesList` function does not include deleted underlying assets.

AAVE V3 allows rewards to continue to be generated and claimed regardless of the active, pause, or deletion status of the underlying asset pool that has registered reward settings. Therefore, if you do not want to provide rewards from a pool that is no longer in use or is temporarily paused, you must manually set `emissionPerSecond` to 0.

RECOMMENDATIONS

Provide a function that allows the administrator to change the reward settings of any asset requested as a parameter, not the return value of the `Pool.getReservesList` function. However, to prevent the weight ratio from being broken, it is recommended to limit it to assets with a weight set to 0.

STATUS Acknowledged

Ozys: The `dropReserve` is possible when the `totalSupply` of aToken, debtToken is 0, which is a situation that rarely occurs. Therefore, we believe there will be no need to use `dropReserve` and will not modify it. In case a certain reserve is changed to pause or inactive, we will first set the reward weight to 0.

● LOW

L-01. LendingRewardTokenManager: A division by zero error can occur when calculating emissionPerSecond Fixed

IMPACT

The `update` function reverts when `totalWeight` is 0. It is impossible to reset all existing weights to 0.

DESCRIPTION

Solidity

```solidity
function update() public onlyOwner {
  IPool pool = IPool(ADDRESSES_PROVIDER.getPool());
  address[] memory reserves = pool.getReservesList();
  RewardsDataTypes.RewardsConfigInput[] memory config = new RewardsDataTypes.RewardsConfigInput[](reserves.length);

  for (uint256 i = 0; i < reserves.length; i++) {
    DataTypes.ReserveData memory reserveData = pool.getReserveData(reserves[i]);
    address _asset = reserveData.aTokenAddress;

    RewardsDataTypes.RewardsConfigInput memory v = RewardsDataTypes.RewardsConfigInput({
@>    emissionPerSecond: uint88(totalRewardPerSecond * weight[reserves[i]] / totalWeight),
      totalSupply: IERC20Detailed(_asset).totalSupply(),
      distributionEnd: type(uint32).max,
      asset: _asset,
      reward: rewardToken,
      transferStrategy: ITransferStrategyBase(transferStrategy),
      rewardOracle: IEACAggregatorProxy(rewardOracle)
    });
    config[i] = v;
  }

  IEmissionManager(EMISSION_MANAGER).configureAssets(config);
}
```

File 3 - periphery/rewards/LendingRewardTokenManager.sol#91-110 Function: update

The update function is automatically called when the previously set weight or totalRewardPerSecond is changed. If there is a situation where you want to reset all weights to 0, the transaction removing the last remaining weight (making `totalWeight` 0) is always reverted due to a division by zero. Therefore, it is impossible to reset all weights to 0.

Solidity
```
function manage(address _asset, uint256 _weight, bool _withUpdate) external onlyOwner {
  // add
  if (weight[_asset] == 0) {
    require(_weight != 0);

    weight[_asset] = _weight;
    totalWeight += _weight;
  } else {
@>  require(_withUpdate);
    totalWeight -= weight[_asset];
    weight[_asset] = _weight;
    totalWeight += _weight;
  }

@>if (_withUpdate) update();
}
```
File 4 - periphery/rewards/LendingRewardTokenManager.sol#74-89 Function: manage

Also, if you call `setTotalRewardPerSecond` without setting a weight, it fails due to division by zero.

Solidity
```
    function setTotalRewardPerSecond(uint256 _totalRewardPerSecond) external onlyOwner {
      totalRewardPerSecond = _totalRewardPerSecond;
  @>  update();
    }
```
File 5 - periphery/rewards/LendingRewardTokenManager.sol#58-61 Function: setTotalRewardPerSecond

RECOMMENDATIONS

Handle exceptions when `totalWeight` is 0 in the calculation of `emissionPerSecond`.

```
function update() public onlyOwner {
  IPool pool = IPool(ADDRESSES_PROVIDER.getPool());
  address[] memory reserves = pool.getReservesList();
  RewardsDataTypes.RewardsConfigInput[] memory config = new RewardsDataTypes.RewardsConfigInput[](reserves.length);

  for (uint256 i = 0; i < reserves.length; i++) {
    DataTypes.ReserveData memory reserveData = pool.getReserveData(reserves[i]);
    address _asset = reserveData.aTokenAddress;

    RewardsDataTypes.RewardsConfigInput memory v = RewardsDataTypes.RewardsConfigInput({
-     emissionPerSecond: uint88(totalRewardPerSecond * weight[reserves[i]] / totalWeight),
+     emissionPerSecond: totalWeight > 0 ? uint88(totalRewardPerSecond * weight[reserves[i]] / totalWeight) : 0,
      totalSupply: IERC20Detailed(_asset).totalSupply(),
      distributionEnd: type(uint32).max,
      asset: _asset,
      reward: rewardToken,
      transferStrategy: ITransferStrategyBase(transferStrategy),
      rewardOracle: IEACAggregatorProxy(rewardOracle)
    });
    config[i] = v;
  }

  IEmissionManager(EMISSION_MANAGER).configureAssets(config);
}
```

At `setTotalRewardPerSecond` function, It is recommended to check `totalWeight` and revert with an explicit error message.

```
 function setTotalRewardPerSecond(uint256 _totalRewardPerSecond) external onlyOwner {
+  require(totalWeight > 0, "totalWeight is 0");
   totalRewardPerSecond = _totalRewardPerSecond;
   update();
 }
```

STATUS Fixed

Ozys: Fixed to prevent division by zero
Fixed in commit `45f3cf1b057db0b1757797d0ae05de6ac44fe0ca`

L-02. LendingRewardTokenManager: Cannot change `transferStrategy`

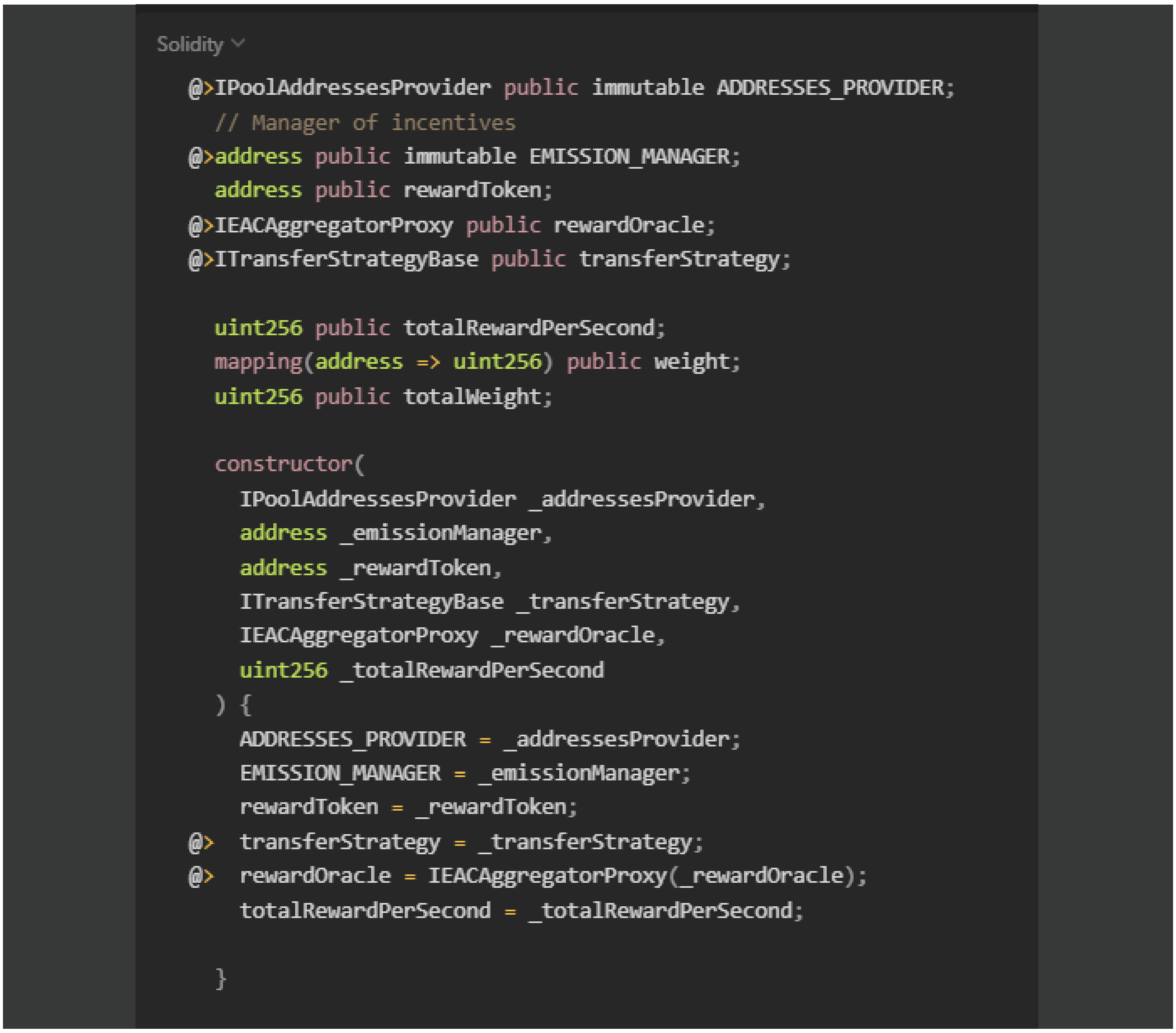
Fixed

IMPACT

Cannot change the `transferStrategy`.

DESCRIPTION

When you call `EmissionManager.setTransferStrategy`, you can change the `transferStrategy` for each reward token. Given that the `transferStrategy` variable is not defined as immutable, it appears to have been designed with the possibility of change.



```
Solidity
@>IPoolAddressesProvider public immutable ADDRESSES_PROVIDER;
  // Manager of incentives
@>address public immutable EMISSION_MANAGER;
  address public rewardToken;
@>IEACAggregatorProxy public rewardOracle;
@>ITransferStrategyBase public transferStrategy;

  uint256 public totalRewardPerSecond;
  mapping(address => uint256) public weight;
  uint256 public totalWeight;

  constructor(
    IPoolAddressesProvider _addressesProvider,
    address _emissionManager,
    address _rewardToken,
    ITransferStrategyBase _transferStrategy,
    IEACAggregatorProxy _rewardOracle,
    uint256 _totalRewardPerSecond
  ) {
    ADDRESSES_PROVIDER = _addressesProvider;
    EMISSION_MANAGER = _emissionManager;
    rewardToken = _rewardToken;
@>  transferStrategy = _transferStrategy;
@>  rewardOracle = IEACAggregatorProxy(_rewardOracle);
    totalRewardPerSecond = _totalRewardPerSecond;

  }
```

File 6 - periphery/rewards/LendingRewardTokenManager.sol#L24-50 Function: constructor

Similar to `transferStrategy`, the `rewardOracle` defined as a general variable and it can be changed. But there is no public function to change the `transferStrategy` and call `EmissionManager.setTransferStrategy`. Therefore, you cannot change the `transferStrategy` in the future.

The PullRewardsTransferStrategyForSwap contract to be registered as a `transferStrategy` makes a contract call to the KlaySwap governance to request a reward. If the governance is changed to a new version in the future, you should be able to change the address of the governance set in PullRewardsTransferStrategyForSwap, or change the `transferStrategy`.

RECOMMENDATIONS

Create a public function to change `transferStrategy` and call `EmissionManager.setTransferStrategy`. If you do not intend to change it, it is recommended to define it as an immutable variable.

STATUS Fixed

Ozys: Added the `setTransferStrategy` function to make it changeable.
Fixed in commit `45f3cf1b057db0b1757797d0ae05de6ac44fe0ca`

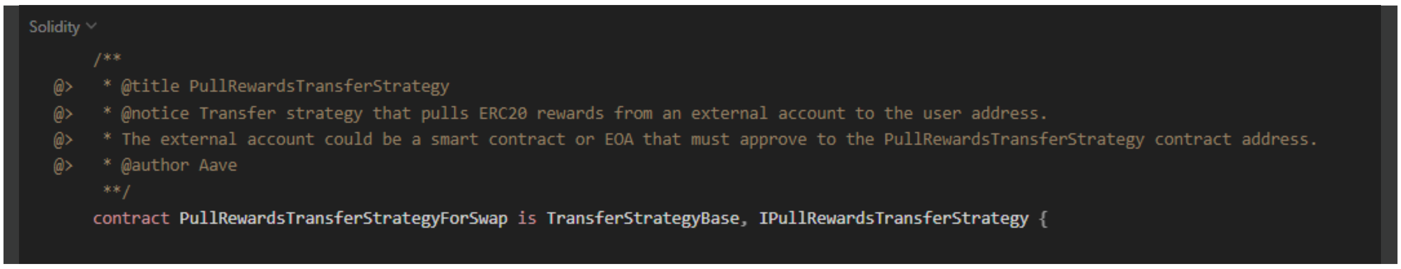
L-03. PullRewardsTransferStrategyForSwap: Incorrect NatSpec comments Fixed

IMPACT

Provides incorrect information in comments.

DESCRIPTION

The title, notice, author information in the NatSpec comments is incorrect.



```solidity
    /**
@>   * @title PullRewardsTransferStrategy
@>   * @notice Transfer strategy that pulls ERC20 rewards from an external account to the user address.
@>   * The external account could be a smart contract or EOA that must approve to the PullRewardsTransferStrategy contract address.
@>   * @author Aave
     **/
    contract PullRewardsTransferStrategyForSwap is TransferStrategyBase, IPullRewardsTransferStrategy {
```

File 7 - periphery/rewards/transfer-strategies/PullRewardsTransferStrategy.sol#L10-16

RECOMMENDATIONS

Correct the comments.

STATUS Fixed

Ozys: Fixed the comments.
Fixed in commit `45f3cf1b057db0b1757797d0ae05de6ac44fe0ca`

ABOUT 78ResearchLab

78ResearchLab is a offensive security corporation offering security auditing, penetration testing, education to enterprises, national organizations, and laboratories with the goal of making safe and convenience digital world. We have our own proprietary technology from system/security analysis and projects on various industries. We are working with the top technical experts who have won prizes in global Hacking Competition/CTF, reported numerous security vulnerabilities, and have 10 years of experience in the information security.
Learn more about us at https://www.78researchlab.com/.

ABOUT RED SPIDER

Red Spider offers cyber security research and auditing service with our new R&D technologies and customized solution in IoT, OS, Web3.
Learn more about red spider at https://www.78researchlab.com/redSpider.html.