Pendle v2 (Part 1, Follow up 1) Audit Report

July 15, 2022

WATCHPUG

Table of Contents

Summary

This report has been prepared for Pendle v2 (Part 1, Follow up 1) Audit Report smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

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

Overview

Project Summary

Project Name	**Pendle v2**
Codebase	**https://github.com/pendle-finance/pendle-core-internal-v2**
Commit	**2be5ad0da54f76a637ddd6a223d49aa02cb8c07f**
Language	**Solidity**

Audit Summary

Delivery Date	**July 15, 2022**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**9**

WP-H1: `PendleYearnVaultScy.sol` Wrong decimals for `exchangeRate()`

High

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/SuperComposableYield/SCY-implementations/PendleYearnVaultScy.sol#L81-L87

```
81      /**
82       * @notice Calculates and updates the exchange rate of shares to underlying asset token
83       * @dev It is the price per share of the yvToken
84       */
85      function exchangeRate() public view override returns (uint256) {
86          return IYearnVault(yvToken).pricePerShare();
87      }
```

Per the EIP-5115 spec, `exchangeRate()` of SCY is supposed to be:

> exchange rate from SCY token amount into asset amount, scaled by a fixed scaling factor of 1e18.

However, Yearn's `Vault.vy` will take the `decimals` of the underlying token as the `decimals` of the `pricePerShare`:

https://github.com/yearn/yearn-vaults/blob/beff27908bb2ae017ed73b773181b9b93f7435ad/contracts/Vault.vy#L1173

```
1165  @view
1166  @external
1167  def pricePerShare() -> uint256:
1168      """
1169      @notice Gives the price for a single Vault share.
1170      @dev See dev note on `withdraw`.
```

```
1171        @return The value of a single share.
1172        """
1173        return self._shareValue(10 ** self.decimals)
```

https://github.com/yearn/yearn-vaults/blob/beff27908bb2ae017ed73b773181b9b93f7435ad/contracts/Vault.vy#L311-L312

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/libraries/SCY/SCYUtils.sol#L7-L17

```
7       function scyToAsset(uint256 exchangeRate, uint256 scyAmount) internal pure
    returns (uint256) {
8           return (scyAmount * exchangeRate) / ONE;
9       }
10
11      function assetToScy(uint256 exchangeRate, uint256 assetAmount)
12          internal
13          pure
14          returns (uint256)
15      {
16          return (assetAmount * ONE) / exchangeRate;
17      }
```

As a result, when the underlying token's decimals is not `18`, `scyToAsset()` and `assetToScy()` will malfunction and the whole `PendleYearnVaultScy` contract will also be malfunctioning.

For example, `yvUSDC`'s `pricePerShare()` will be about `1e6` instead of the expected `1e18`.

Recommendation

Read and save `IYearnVault.decimals` in the constructor function.

Status

✓ Fixed

WP-L2: `PendleYearnVaultScy.sol` decimals of the SCY token should not be fixed as `18`

Low

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/SuperComposableYield/SCY-implementations/PendleYearnVaultScy.sol#L7-L20

```
7   contract PendleYearnVaultSCY is SCYBase {
8       address public immutable underlying;
9       address public immutable yvToken;
10
11      constructor(
12          string memory _name,
13          string memory _symbol,
14          address _yvToken
15      ) SCYBase(_name, _symbol, _yvToken) {
16          require(_yvToken != address(0), "zero address");
17          yvToken = _yvToken;
18          underlying = IYearnVault(yvToken).token();
19          _safeApprove(underlying, yvToken, type(uint256).max);
20      }
```

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/SuperComposableYield/base-implementations/SCYBase.sol#L12-L23

```
12  abstract contract SCYBase is ISuperComposableYield, PendleERC20, TokenHelper {
13      using Math for uint256;
14
15      address public immutable yieldToken;
16
17      constructor(
18          string memory _name,
19          string memory _symbol,
20          address _yieldToken
```

```
21      ) PendleERC20(_name, _symbol, 18) {
22          yieldToken = _yieldToken;
23      }
```

> Decimals of the SCY token should reflect the underlying GYGP's accounting asset's `decimals`

The `decimals` of `PendleYearnVaultScy.sol` is now fixed as `18`, while the yvToken's decimals can be different.

Status

✓ Fixed

WP-L2: `PendleQiTokenSCY.sol` decimals of the SCY token should not be fixed as `18`

Low

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/SuperComposableYield/SCY-implementations/BenQi/PendleQiTokenSCY.sol#L12-L28

```
12  contract PendleQiTokenSCY is SCYBaseWithRewards, PendleQiTokenHelper {
13      address public immutable underlying;
14      address public immutable QI;
15      address public immutable WAVAX;
16      address public immutable comptroller;
17      address public immutable qiToken;
18
19      constructor(
20          string memory _name,
21          string memory _symbol,
22          address _qiToken,
23          address _WAVAX,
24          uint256 _initialExchangeRateMantissa
25      )
26          SCYBaseWithRewards(_name, _symbol, _qiToken)
27          PendleQiTokenHelper(_qiToken, _initialExchangeRateMantissa)
28      {
```

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/SuperComposableYield/base-implementations/SCYBaseWithRewards.sol#L11-L21

```
11  abstract contract SCYBaseWithRewards is SCYBase, RewardManager {
12      using Math for uint256;
13      using ArrayLib for address[];
14
15      constructor(
16          string memory _name,
```

```
17          string memory _symbol,
18          address _yieldToken
19      )
20          SCYBase(_name, _symbol, _yieldToken) // solhint-disable-next-line
    no-empty-blocks
21      {}
```

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/SuperComposableYield/base-implementations/SCYBase.sol#L12-L23

```
12  abstract contract SCYBase is ISuperComposableYield, PendleERC20, TokenHelper {
13      using Math for uint256;
14
15      address public immutable yieldToken;
16
17      constructor(
18          string memory _name,
19          string memory _symbol,
20          address _yieldToken
21      ) PendleERC20(_name, _symbol, 18) {
22          yieldToken = _yieldToken;
23      }
```

> Decimals of the SCY token should reflect the underlying GYGP's accounting asset's `decimals`

The `decimals` of `PendleQiTokenSCY.sol` is now fixed as `18`, while the QiToken's decimals is `8`.

Status

✓ Fixed

WP-H3: `PendleYearnVaultScy.sol` Wrong implementation of `_redeem()`

High

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/SuperComposableYield/SCY-implementations/PendleYearnVaultScy.sol#L56-L75

```
56      function _redeem(address tokenOut, uint256 amountSharesToRedeem)
57          internal
58          virtual
59          override
60          returns (uint256 amountTokenOut)
61      {
62          if (tokenOut == yvToken) {
63              amountTokenOut = amountSharesToRedeem;
64          } else {
65              // tokenOut == underlying
66              uint256 sharesRedeemed =
    IYearnVault(yvToken).withdraw(amountSharesToRedeem);
67
68              require(
69                  sharesRedeemed != amountSharesToRedeem,
70                  "Yearn Vault SCY: Not allowed to redeem all shares"
71              );
72
73              amountTokenOut = _selfBalance(underlying);
74          }
75      }
```

https://github.com/yearn/yearn-vaults/blob/efb47d8a84fcb13ceebd3ceb11b126b323bcc05d/contracts/Vault.vy#L1054

```
1054    @return The quantity of tokens redeemed for `_shares`.
```

1. `IYearnVault.withdraw()` returns the **quantity of tokens redeemed** for `_shares`, not the `sharesRedeemed`.
2. The `require` condition is wrong: `require(sharesRedeemed != amountSharesToRedeem)` should be `require(sharesRedeemed == amountSharesToRedeem)`.

Status

✓ **Fixed**

WP-M4: `PendleYieldToken.sol` transfer of YT tokens may revert or cause loss of rewards after expiration

Medium

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/core/YieldContracts/PendleYieldToken.sol#L343-L350

```
343  function _beforeTokenTransfer(
344      address from,
345      address to,
346      uint256
347  ) internal override {
348      _updateAndDistributeRewardsForTwo(from, to);
349      _distributeInterestForTwo(from, to);
350  }
```

The `_beforeTokenTransfer` hook on `YT` will distribute rewards for both `from` and `to`:

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/libraries/RewardManagerAbstract.sol#L27-L35

```
27  function _updateAndDistributeRewardsForTwo(address user1, address user2) internal
    virtual {
28      (address[] memory tokens, uint256[] memory indexes) = _updateRewardIndex();
29      if (tokens.length == 0) return;
30
31      if (user1 != address(0) && user1 != address(this))
32          _distributeRewardsPrivate(user1, tokens, indexes);
33      if (user2 != address(0) && user2 != address(this))
34          _distributeRewardsPrivate(user2, tokens, indexes);
35  }
```

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/core/YieldContracts/PendleYieldToken.sol#L327-L340

```
327  function _updateRewardIndex()
328      internal
329      override
330      returns (address[] memory tokens, uint256[] memory indexes)
331  {
332      tokens = getRewardTokens();
333      if (isExpired()) {
334          indexes = new uint256[](tokens.length);
335          for (uint256 i = 0; i < tokens.length; i++)
336              indexes[i] = postExpiry.firstRewardIndex[tokens[i]];
337      } else {
338          indexes = ISuperComposableYield(SCY).rewardIndexesCurrent();
339      }
340  }
```

When the `YT` is expired, the reward indexes will be read from `postExpiry`.

However, `PostExpiryData` will only be updated when someone calls a function with the `updateData` modifier, including:

- `mintPY()`
- `redeemPY()`
- `redeemDueInterestAndRewards()`
- `redeemRewardsPostExpiryForTreasury()`

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/core/YieldContracts/PendleYieldToken.sol#L50-L54

```
50  modifier updateData() {
51      if (isExpired()) _setPostExpiryData();
52      _;
53      _updateScyReserve();
54  }
```

The a YT holder tries to transfer it after expiration but before anyone calls the functions above, `_updateRewardIndex` will return `indexes` from empty `postExpiry`, which means the index will be `0`.

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/libraries/RewardManagerAbstract.sol#L38-L68

```
38  function _distributeRewardsPrivate(
39      address user,
40      address[] memory tokens,
41      uint256[] memory indexes
42  ) private {
43      assert(user != address(0) && user != address(this));
44
45      uint256 userShares = _rewardSharesUser(user);
46
47      for (uint256 i = 0; i < tokens.length; ++i) {
48          address token = tokens[i];
49          uint256 index = indexes[i];
50          uint256 userIndex = userReward[token][user].index;
51
52          if (userIndex == 0) {
53              userReward[token][user].index = index.Uint128();
54              continue;
55          }
56
57          if (userIndex == index) continue;
58
59          uint256 deltaIndex = index - userIndex;
60          uint256 rewardDelta = userShares.mulDown(deltaIndex);
61          uint256 rewardAccrued = userReward[token][user].accrued + rewardDelta;
62
63          userReward[token][user] = UserReward({
64              index: index.Uint128(),
65              accrued: rewardAccrued.Uint128()
66          });
67      }
68  }
```

If `userIndex` > 0, then the transaction will revert due to underflow; if the `userIndex` is 0 (the

user has never settled the rewards before), then the transcation will go though, and `rewardDelta` is 0, which means the sender will lose the rewards.

Recommendation

Change `_updateRewardIndex()` to:

```
327  function _updateRewardIndex()
328      internal
329      override
330      returns (address[] memory tokens, uint256[] memory indexes)
331  {
332      tokens = getRewardTokens();
333      if (isExpired()) {
334          _setPostExpiryData();
335          indexes = new uint256[](tokens.length);
336          for (uint256 i = 0; i < tokens.length; i++)
337              indexes[i] = postExpiry.firstRewardIndex[tokens[i]];
338      } else {
339          indexes = ISuperComposableYield(SCY).rewardIndexesCurrent();
340      }
341  }
```

A similar issue also applies to `_distributeInterestForTwo()`, and it will be fixed once the fixed above is applied, as `_updateAndDistributeRewardsForTwo()` will be called before `_distributeInterestForTwo()` and `_setPostExpiryData` if needed.

Status

✓ Fixed

WP-L5: `ActionSCYAndYTBase.sol#_swapScyForExactYt()` Wrong return value

Low

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/core/actions/base/ActionSCYAndYTBase.sol#L112-L127

```
112     function _swapScyForExactYt(
113         address receiver,
114         address market,
115         uint256 exactYtOut,
116         uint256 maxScyIn
117     ) internal returns (uint256 netScyIn) {
118         (, , IPYieldToken YT) = IPMarket(market).readTokens();
119 
120         IPMarket(market).swapExactPtForScy(
121             address(YT),
122             exactYtOut, // exactPtIn = exactYtOut
123             _encodeSwapScyForExactYt(msg.sender, receiver, maxScyIn)
124         );
125 
126         emit SwapYTAndSCY(receiver, exactYtOut.Int(), netScyIn.neg());
127     }
```

`netScyIn` is never set within the function, therefore it will always be `0`.

Status

✓ Fixed

WP-L6: `ActionCallback._callbackSwapScyForExactYt()` Wrong implementation to guard against precision issue while calculating `totalScyNeed`

Low

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/47470947c7d9c4d9f7bc9efdc549b12059d95cdc/contracts/core/actions/ActionCallback.sol#L79-L113

```
79  /// @dev refer to _swapScyForExactYt
80  function _callbackSwapScyForExactYt(
81      address market,
82      int256 ptToAccount,
83      int256 scyToAccount,
84      bytes calldata data
85  ) internal {
86      VarsSwapScyForExactYt memory vars;
87      (vars.payer, vars.receiver, vars.maxScyToPull) =
    _decodeSwapScyForExactYt(data);
88      (ISuperComposableYield SCY, , IPYieldToken YT) =
    IPMarket(market).readTokens();
89
90      /// ------------------------------------------------------------
91      /// calc totalScyNeed
92      /// ------------------------------------------------------------
93      SCYIndex scyIndex = SCY.newIndex();
94      uint256 ptOwed = ptToAccount.abs();
95      uint256 totalScyNeed = scyIndex.assetToScy(ptOwed);
96      // to guard against precision issue of lacking a few units of SCY. rawDivUp is
    not Fixed-Point division
97      totalScyNeed += SCYIndex.unwrap(scyIndex).rawDivUp(SCYUtils.ONE);
98
99      /// ------------------------------------------------------------
100     /// calc netScyToPull
101     /// ------------------------------------------------------------
102     uint256 scyReceived = scyToAccount.Uint();
103     uint256 netScyToPull = totalScyNeed.subMax0(scyReceived);
```

```
104         require(netScyToPull <= vars.maxScyToPull, "exceed SCY in limit");
105
106         /// ------------------------------------------------------------
107         /// mint & transfer
108         /// ------------------------------------------------------------
109         SCY.safeTransferFrom(vars.payer, address(YT), netScyToPull);
110
111         uint256 amountPYout = YT.mintPY(market, vars.receiver);
112         require(amountPYout >= ptOwed, "insufficient pt to pay");
113     }
```

At L95, `totalScyNeed` is calculated as:

$$totalScyNeed = \frac{ptOwed \cdot 10^{18}}{exchangeRate_{scy}}$$

The result of the div at L95 should be used directly without any addition when there is no precision loss

For instance:

Given:

- $exchangeRate_{scy}$: 1.1e18
- $ptOwed$: 11e18

Expected results:

- As result of the div at L95 is `10e18`, with no remain (no precision loss), therefore, it should not add `totalScyNeed`
- The final value of `totalScyNeed` should be `10e18`

Actual results:

- L97 will add $exchangeRate_{scy}.divUp(10^{18}) = (1.1 \times 10^{18}).divUp(10^{18}) = 2$ to `totalScyNeed`
- The final value of `totalScyNeed` is 10e18 + 2

When there is a precision loss for the div at L95, the remain 1 wei should be added to `totalScyNeed`

For instance:

Given:

- $exchangeRate_{scy}$: 1.1e18
- $ptOwed$: 10e18

Expected results:

- As the result of the div at L95 is 9090909090909090909, there is a precision loss of 0.090909090909090909... wei, therfore, `totalScyNeed` should add 1 wei to fix the precision loss
- The final value of `totalScyNeed` should be `9090909090909090909 + 1 = 9090909090909090910`

Actual results:

- L97 adds $exchangeRate_{scy}.divUp(10^{18}) = (1.1 \times 10^{18}).divUp(10^{18}) = 2$ to `totalScyNeed`
- The final value of `totalScyNeed` is `9090909090909090909 + 2 = 9090909090909090911`

Recommendation

Consider adding a function named `assetToScyUp()` :

$$scyAmount = (assetAmount \cdot 10^{18}).divUp(exchangeRate_{scy})$$

```
function assetToScyUp(uint256 exchangeRate, uint256 assetAmount)
    internal
    pure
    returns (uint256)
{
    return (assetAmount * ONE).rawDivUp(exchangeRate);
}
```

Status

✓ Fixed

WP-H7: `_swapScyForExactYt()` When current `exchangeRate` is lower than the highest history `exchangeRate`, the user will be charged for extra SCY and lose part of the PT minted to the market

High

Issue Description

`scyIndexCurrent()` is the highest history exchangeRate for the SCY:

https://github.com/pendle-finance/pendle-core-internal-v2/blob/47470947c7d9c4d9f7bc9efdc549b12059d95cdc/contracts/core/YieldContracts/PendleYieldToken.sol#L175-L179

```
175  /// @dev maximize the current rate with the previous rate to guarantee
     non-decreasing rate
176  function scyIndexCurrent() public returns (uint256 currentIndex) {
177      currentIndex = Math.max(ISuperComposableYield(SCY).exchangeRate(),
     _scyIndexStored);
178      _scyIndexStored = currentIndex.Uint128();
179  }
```

As a result, when the current exchangeRate is lower than the highest history exchangeRate (`scyIndexCurrent()`), at `ActionCallback._callbackSwapScyForExactYt()` L111, `YT.mintPY(market, vars.receiver)` will mint extra some PT to the PtMarket.

`PendleYieldToken.mintPY()` is using `PendleYieldToken.scyIndexCurrent()` to calculate the amounts of PT and YT tokens that can be minted with `scyAmountIn` of SCY.

However, `PendleRouter.swapScyForExactYt()` is using `scy.exchangeRate()` (the current exchange) to calculate the `totalScyNeed` for the desired amount of YT:

$$totalScyNeed = \frac{exactYtOut \cdot 10^{18}}{exchangeRate_{scy}}$$

$exchangeRate_{scy}$ is `SCY.exchangeRate()` , while L111 `YT.mintPY(market, vars.receiver)` is using `PendleYieldToken.scyIndexCurrent()` , highest history exchangeRate of the SCY.

Therefore, when $exchangeRate_{scy}$ < `scyIndexCurrent()` , the `totalScyNeed` calculated will be higher than expected, so that L111 `YT.mintPY(market, vars.receiver)` will mint more PT than expected to the `market` , which constitutes user's loss.

https://github.com/pendle-finance/pendle-core-internal-v2/blob/47470947c7d9c4d9f7bc9efdc549b12059d95cdc/contracts/core/actions/ActionCallback.sol#L79-L113

```
79      /// @dev refer to _swapScyForExactYt
80      function _callbackSwapScyForExactYt(
81          address market,
82          int256 ptToAccount,
83          int256 scyToAccount,
84          bytes calldata data
85      ) internal {
86          VarsSwapScyForExactYt memory vars;
87          (vars.payer, vars.receiver, vars.maxScyToPull) =
    _decodeSwapScyForExactYt(data);
88          (ISuperComposableYield SCY, , IPYieldToken YT) =
    IPMarket(market).readTokens();
89
90          /// ------------------------------------------------------------
91          /// calc totalScyNeed
92          /// ------------------------------------------------------------
93          SCYIndex scyIndex = SCY.newIndex();
94          uint256 ptOwed = ptToAccount.abs();
95          uint256 totalScyNeed = scyIndex.assetToScy(ptOwed);
96          // to guard against precision issue of lacking a few units of SCY.
    rawDivUp is not Fixed-Point division
97          totalScyNeed += SCYIndex.unwrap(scyIndex).rawDivUp(SCYUtils.ONE);
98
99          /// ------------------------------------------------------------
100         /// calc netScyToPull
101         /// ------------------------------------------------------------
102         uint256 scyReceived = scyToAccount.Uint();
103         uint256 netScyToPull = totalScyNeed.subMax0(scyReceived);
104         require(netScyToPull <= vars.maxScyToPull, "exceed SCY in limit");
105
106         /// ------------------------------------------------------------
```

```
107         /// mint & transfer
108         /// ------------------------------------------------------------
109         SCY.safeTransferFrom(vars.payer, address(YT), netScyToPull);
110
111         uint256 amountPYout = YT.mintPY(market, vars.receiver);
112         require(amountPYout >= ptOwed, "insufficient pt to pay");
113     }
```

Recommendation

Consider using `scyIndexCurrent()` instead of `exchangeRate()` to calculate `totalScyNeed` in `_callbackSwapScyForExactYt()` .

Or, send the extra PT back to the `msg.sender` (`vars.receiver`).

Status

✓ Fixed

WP-G8: Avoid unnecessary expensive external calls can save gas

Gas

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/27da8ab658a0ba2482a1695a5da1f992f6dafd3d/contracts/core/YieldContracts/PendleYieldToken.sol#L275-L293

```
275  function _doTransferOutRewards(address user, address receiver)
276      internal
277      virtual
278      override
279      returns (uint256[] memory rewardAmounts)
280  {
281      address[] memory tokens = getRewardTokens();
282
283      if (isExpired()) {
284          // post-expiry, all incoming rewards will go to the treasury
285          // hence, we can save users one _redeemExternal here
286          for (uint256 i = 0; i < tokens.length; i++)
287              postExpiry.userRewardOwed[tokens[i]] -=
     userReward[tokens[i]][user].accrued;
288          rewardAmounts = __doTransferOutRewardsLocal(tokens, user, receiver);
289      } else {
290          _redeemExternalReward();
291          rewardAmounts = __doTransferOutRewardsLocal(tokens, user, receiver);
292      }
293  }
```

In the current implementation, `_redeemExternalReward()` will be called whenever `redeemDueInterestAndRewards()` is called.

However, `ISuperComposableYield(SCY).claimRewards()` is a quite expensive call. If someone has already called `redeemDueInterestAndRewards` recently (since once they do, they actually harvested the rewards for the whole SCY), there is a good chance that the contract itself already has sufficient balance for the claim.

Furthermore, it's unnecessary to transfer fees to the treasury every time.

Consider adding a new function called `__doTransferOutRewardsWithRedeemExternalReward()` :

```
function __doTransferOutRewardsWithRedeemExternalReward(
    address[] memory tokens,
    address user,
    address receiver
) internal returns (uint256[] memory rewardAmounts) {
    uint256 feeRate = IPYieldContractFactory(factory).rewardFeeRate();

    rewardAmounts = new uint256[](tokens.length);

    bool externalRewardRedeemed;
    for (uint256 i = 0; i < tokens.length; i++) {
        uint256 rewardPreFee = userReward[tokens[i]][user].accrued;

        userReward[tokens[i]][user].accrued = 0;

        uint256 feeAmount = rewardPreFee.mulDown(feeRate);
        rewardAmounts[i] = rewardPreFee - feeAmount;

        // NEWLY ADDED STORAGE: treasuryFee
        // instead of push funds to the treasury for every call
        // the admin will need to pull funds; this saves gas
        treasuryFee[tokens[i]] += feeAmount;

        //  redeemExternalReward only if no enough funds
        if (!externalRewardRedeemed && IERC20(tokens[i]).balanceOf(address(this))
< rewardAmounts[i]) {
            _redeemExternalReward();
            externalRewardRedeemed = true;
        }
        _transferOut(tokens[i], receiver, rewardAmounts[i]);
    }
}
```

Status

✓ Fixed

Appendix

Timeliness of content

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

Disclaimer

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