Lorenzo PendleLorenzoSUsd1PlusSY Audit Report

Oct 9, 2025

Table of Contents

Summary

This report has been prepared for Lorenzo 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	**Lorenzo**
Codebase	**https://github.com/Lorenzo-Protocol/Pendle-SY-Public**
Commit	**5b81f9d24b17b1d26a47429f5cb7b71d0237dc2f**
Language	**Solidity**

Audit Summary

Delivery Date	**Oct 9, 2025**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**9**

[WP-H1] `_deposit` causes SY to be unable to receive shares because `sUSD1PlusVault` requires the depositor (SY) to call `confirmShare` to mint shares.

High

Issue Description

The standard `deposit` interface is divided into 2 steps: first, set the share to pending status; then after the depositor (SY) obtains an off-chain signature, the depositor calls `confirmShare` to confirm.

Currently, SY does not have a place to call `confirmShare` .

https://github.com/Lorenzo-Protocol/Pendle-SY-Public/blob/85b59e61e4d22867a872de078708d3b75729d86d/contracts/core/StandardizedYield/implementations/Lorenzo/PendleLorenzoSUsd1PlusSY.sol#L44-L52

```
44      function _deposit(address tokenIn, uint256 amountDeposited) internal virtual override returns (uint256) {
45          if (!isValidTokenIn(tokenIn)) {
46              revert InvalidTokenIn(tokenIn);
47          }
48
49          uint256 preBalance = _selfBalance(sUSD1PlusVaultAddr);
50          ISUsd1PlusVault(sUSD1PlusVaultAddr).deposit(tokenIn, amountDeposited);
51          return _selfBalance(sUSD1PlusVaultAddr) - preBalance;
52      }
```

https://bscscan.com/address/0x103e63266f9a7b05b833f4964880a7ece5407968#code#F34#L294

```
293     // @dev deposit underlying asset to the vault, mint shares to the target address
294     function deposit(address underlyingToken, uint256 underlyingAmount)
295         public
296         whenNotPaused
297         notBlacklisted
298         lock
```

```
299         payable
300     {
301         require(acceptUnderlying[underlyingToken] || underlyingToken == underlying
    || underlyingToken == NATIVE_TOKEN, "not accept underlying");
302
303         (uint256 sharesAmount, uint256 unitNav) = onDepositUnderlying(msg.sender,
    underlyingToken, underlyingAmount);
304
305         // emit enough information for accounting
306         address[] memory p = portfolios();
307         uint256[] memory w = weights();
308         emit Deposited(
309             msg.sender,
310             underlyingToken,
311             underlyingAmount,
312             sharesAmount,
313             unitNav,
314             settlementPeriod,
315             p,
316             w,
317             block.timestamp
318         );
319     }
```

https://bscscan.com/address/0x103e63266f9a7b05b833f4964880a7ece5407968#code#F1#L161

```
140     // to handle the depoisted underlying asset
141     function onDepositUnderlying(address from, address underlyingToken, uint256
    underlyingAmount)
142         internal
143         override
144         returns (uint256 sharesAmount, uint256 unitNav)
145     {
146         require(msg.value == 0, "not allowed to deposit native token");
147         // transfer the underlying asset to the ceff wallet
148         address ceffWallet = _portfolios.at(0);
149         IERC20 assetToken = IERC20(underlyingToken);
150         SafeERC20.safeTransferFrom(assetToken, from, ceffWallet,
    underlyingAmount);
151
152         uint256 alignedAmount = alignDepositAmount(underlyingAmount,
    underlyingToken);
```

```
153         require(alignedAmount >= minDepositAmount, "deposit amount too small");
154
155         // calculate the shares amount
156         unitNav = getUnitNav(settlementPeriod);
157         require(unitNav > 0, "oops: zero unit nav");
158         sharesAmount = alignedAmount * Precision / unitNav;
159
160         // calculate the pending shares
161         pendingShares[from][settlementPeriod] += sharesAmount;
162
163         // emit the event
164         emit PendingShares(from, settlementPeriod, sharesAmount);
165     }
166
```

```
176     function confirmShare(
177         uint128 period,
178         bytes32 txHash,
179         uint256 sharesAmount,
180         bytes memory signature
181     )
182         public
183         whenNotPaused
184     {
185         require(sharesAmount > 0, "Invalid shares amount");
186         require(pendingShares[msg.sender][period] >= sharesAmount, "not enough
    shares amount");
187         require(!confirmedTx[txHash], "tx already confirmed");
188         require(!refundedTx[txHash], "tx already refunded");
189
190         // verify the signature
191         bytes32 hash = _hashTypedDataV4(keccak256(abi.encode(
192             CONFIRM_SHARE,
193             period,
194             txHash,
195             msg.sender,
196             sharesAmount
197         )));
198         address signer = ECDSA.recover(hash, signature);
199         require(signers[signer], "Invalid signer");
200         confirmedTx[txHash] = true;
```

```
201
202         // sub the pending shares
203         pendingShares[msg.sender][period] -= sharesAmount;
204         // add the confirmed shares
205         confirmedShares[msg.sender][period] += sharesAmount;
206
207         // mint the shares
208         _mint(msg.sender, sharesAmount);
209
210         // emit the event
211         emit ConfirmShares(msg.sender, txHash, period, sharesAmount);
212     }
```

Recommendation

We noticed that `sUSD1PlusVault` has a privileged function: `directDeposit`, with the comment "method for Pendle or someone else to deposit without KYT."

The privileged function should be used directly for deposits:

```
288     // @dev method for Pendle or someone else to deposit without KYT.
289     // just composit the deposit and confirm function.
290     function directDeposit(address underlyingToken, uint256 underlyingAmount)
    public onlyDepositor {
291         require(acceptUnderlying[underlyingToken] || underlyingToken ==
    underlying);
292                 // transfer the underlying asset to the ceff wallet
293         address ceffWallet = _portfolios.at(0);
294         IERC20 assetToken = IERC20(underlyingToken);
295         SafeERC20.safeTransferFrom(assetToken, msg.sender, ceffWallet,
    underlyingAmount);
296
297         uint256 alignedAmount = alignDepositAmount(underlyingAmount,
    underlyingToken);
298         require(alignedAmount >= minDepositAmount, "deposit amount too small");
299
300         // calculate the shares amount
301         uint256 unitNav = getUnitNav(settlementPeriod);
302         require(unitNav > 0, "oops: zero unit nav");
303         uint256 sharesAmount = alignedAmount * Precision / unitNav;
304
```

```
305         // mint the shares
306         _mint(msg.sender, sharesAmount);
307 
308         // emit the event
309         emit DirectDeposit(msg.sender, settlementPeriod, underlyingToken,
    underlyingAmount, sharesAmount);
310     }
```

Status

✓ Fixed

[WP-M2] `yield` and `asset` should not be the same

Medium

Issue Description

Current implementation has both `yield` and `asset` as `_sUSD1Plus` , but with an exchangeRate not equal to 1e18.

Based on the context, we assume `USD1` should be the asset.

```
getCurrentUnitNav() => 1009260120000000000
```

```
20      constructor(
21          address _usd1,
22          address _usdt,
23          address _usdc,
24          address _sUSD1Plus,
25          address _rewardToken
26      )
27          SYBaseUpg(_sUSD1Plus)
28      {
29          usd1Addr = _usd1;
30          usdtAddr = _usdt;
31          usdcAddr = _usdc;
32
33          sUSD1PlusVaultAddr = _sUSD1Plus;
34          rewardManagerAddr = _rewardToken;
35
36      }
37
    @@ 38,65 @@
66
67      function exchangeRate() public view virtual override returns (uint256) {
68          return ISUsd1PlusVault(sUSD1PlusVaultAddr).getCurrentUnitNav();
69      }
70
```

```
@@ 71,121 @@

    function assetInfo() external view returns (AssetType assetType, address assetAddress, uint8 assetDecimals) {
        return (AssetType.TOKEN, sUSD1PlusVaultAddr, IERC20Metadata(sUSD1PlusVaultAddr).decimals());
    }
```

Status

✓ Fixed

[WP-M3] `getTokensIn()` token is not approved to `_sUSD1Plus`

Medium

Issue Description

Deposit will fail as the transferFrom will revert due to insufficient allowance.

```
141  function onDepositUnderlying(address from, address underlyingToken, uint256
     underlyingAmount)
142      internal
143      override
144      returns (uint256 sharesAmount, uint256 unitNav)
145  {
146      require(msg.value == 0, "not allowed to deposit native token");
147      // transfer the underlying asset to the ceff wallet
148      address ceffWallet = _portfolios.at(0);
149      IERC20 assetToken = IERC20(underlyingToken);
150      SafeERC20.safeTransferFrom(assetToken, from, ceffWallet, underlyingAmount);
151
152      uint256 alignedAmount = alignDepositAmount(underlyingAmount, underlyingToken);
153      require(alignedAmount >= minDepositAmount, "deposit amount too small");
154
155      // calculate the shares amount
156      unitNav = getUnitNav(settlementPeriod);
157      require(unitNav > 0, "oops: zero unit nav");
158      sharesAmount = alignedAmount * Precision / unitNav;
159
160      // calculate the pending shares
161      pendingShares[from][settlementPeriod] += sharesAmount;
162
163      // emit the event
164      emit PendingShares(from, settlementPeriod, sharesAmount);
165  }
```

Status

✓ Fixed

[WP-L4] No need to check if a token is whitelisted in the SY implementation since `SYBase` already checks `isValidTokenIn` .

Low

Issue Description

Consider removing the corresponding check and error definition.

```
40  function _deposit(address tokenIn, uint256 amountDeposited) internal virtual
    override returns (uint256) {
41      if (!isValidTokenIn(tokenIn)) {
42          revert InvalidTokenIn(tokenIn);
43      }
44
45      uint256 preBalance = _selfBalance(sUSD1PlusVaultAddr);
46      ISUsd1PlusVault(sUSD1PlusVaultAddr).deposit(tokenIn, amountDeposited);
47      return _selfBalance(sUSD1PlusVaultAddr) - preBalance;
48  }
```

```
18  error InvalidTokenIn(address tokenIn);
```

```
46  function deposit(
47      address receiver,
48      address tokenIn,
49      uint256 amountTokenToDeposit,
50      uint256 minSharesOut
51  ) external payable nonReentrant returns (uint256 amountSharesOut) {
52      if (!isValidTokenIn(tokenIn)) revert Errors.SYInvalidTokenIn(tokenIn);
53      if (amountTokenToDeposit == 0) revert Errors.SYZeroDeposit();
54
55      _transferIn(tokenIn, msg.sender, amountTokenToDeposit);
56
57      amountSharesOut = _deposit(tokenIn, amountTokenToDeposit);
58      if (amountSharesOut < minSharesOut) revert
    Errors.SYInsufficientSharesOut(amountSharesOut, minSharesOut);
59
60      _mint(receiver, amountSharesOut);
```

```
61      emit Deposit(msg.sender, receiver, tokenIn, amountTokenToDeposit,
    amountSharesOut);
62  }
```

Status

✓ Fixed

[WP-L5] `PendleLorenzoSUsd1PlusReward.getRewardToken()` and public contract variable `rewardToken` 's auto-generated `rewardToken()` are duplicates.

Low

Issue Description

https://docs.soliditylang.org/en/v0.8.30/contracts.html#state-variable-visibility

> Public state variables differ from internal ones only in that the compiler automatically generates getter functions for them, which allows other contracts to read their values.

Consider either using `rewardToken()` and removing `getRewardToken()` , or changing `rewardToken` to internal or private visibility.

https://github.com/Lorenzo-Protocol/Pendle-SY-Public/blob/85b59e61e4d22867a872de078708d3b75729d86d/contracts/core/StandardizedYield/implementations/Lorenzo/PendleLorenzoSUsd1PlusReward.sol#L10-L116

```
10  contract PendleLorenzoSUsd1PlusReward is UUPSUpgradeable, OwnableUpgradeable,
    ISUsd1PlusReward {
11
    @@ 12,27 @@
28
29      // Constants
30      IERC20 public immutable rewardToken;
31
    @@ 32,110 @@
111
112     // @dev get the reward token
113     function getRewardToken() external view override returns (address) {
114         return address(rewardToken);
115     }
116 }
```

Status

✓ Fixed

[WP-I6] `SY._getRewardTokens()` should not include the yield token held by SY, otherwise the yield might be mistakenly distributed as rewards.

Informational

Issue Description

Consider saving `rewardManagerAddr.rewardToken()` as an immutable variable in `PendleLorenzoSUsd1PlusSY` and require `rewardToken != _sUSD1Plus`

Both `PendleLorenzoSUsd1PlusSY.rewardManagerAddr` and `PendleLorenzoSUsd1PlusReward.rewardToken` are immutable and are determined during the deployment of PendleLorenzoSUsd1PlusSY with minimal likelihood of modification.

Based on this, we can add the following directly in `PendleLorenzoSUsd1PlusSY.constructor()` :

```
        rewardToken = rewardManagerAddr.rewardToken();
        require(rewardToken != _sUSD1Plus, ...);
```

This ensures that `SY._getRewardTokens()` is not the `_sUSD1Plus` held by PendleLorenzoSUsd1PlusSY.

This also saves gas consumption in the regular business function `SY._getRewardTokens()` .

https://github.com/Lorenzo-Protocol/Pendle-SY-Public/blob/85b59e61e4d22867a872de078708d3b75729d86d/contracts/core/StandardizedYield/implementations/Lorenzo/PendleLorenzoSUsd1PlusSY.sol#L9-L126

```
 9  contract PendleLorenzoSUsd1PlusSY is UUPSUpgradeable, SYBaseWithRewardsUpg {
10      using PMath for uint256;
11
12      address public immutable sUSD1PlusVaultAddr;
13      address public immutable rewardManagerAddr;
    @@ 14,18 @@
```

```
19
    @@ 20,73 @@
74
75      /**
76       * @dev See {IStandardizedYield-getRewardTokens}
77       */
78      function _getRewardTokens() internal view override returns (address[] memory)
    {
79          return
    ArrayLib.create(ISUsd1PlusReward(rewardManagerAddr).getRewardToken());
80      }
81
82      function _redeemExternalReward() internal override {
83          ISUsd1PlusReward(rewardManagerAddr).claimRewards(address(this));
84      }
85
    @@ 86,125 @@
126 }
```

```
10  contract PendleLorenzoSUsd1PlusReward is UUPSUpgradeable, OwnableUpgradeable,
    ISUsd1PlusReward {
11
    @@ 12,27 @@
28
29      // Constants
30      IERC20 public immutable rewardToken;
31
    @@ 32,110 @@
111
112     // @dev get the reward token
113     function getRewardToken() external view override returns (address) {
114         return address(rewardToken);
115     }
116 }
```

Status

✓ Fixed

[WP-I7] Current implementation does not support direct yield token (sUSD1Plus) deposits.

Informational

Issue Description

Common Streaming Yield (SY) tokens support direct yield token deposits.

```
115  function isValidTokenIn(address token) public view virtual override returns (bool)
     {
116      return token == usd1Addr || token == usdtAddr || token == usdcAddr;
117  }
```

Status

✓ Fixed

[WP-I8] `PendleLorenzoSUsd1PlusReward.releaseRewards()` should use `rewardToken.safeTransferFrom(msg.sender, address(this), amount)` to transfer rewards from the caller to the contract.

Informational

Issue Description

Alternatively, the rewards must be transferred to the rewards contract through other valid methods.

The docs didn't specify how the reward tokens are sent to the contract.

https://github.com/Lorenzo-Protocol/Pendle-SY-Public/blob/85b59e61e4d22867a872de078708d3b75729d86d/contracts/core/StandardizedYield/implementations/Lorenzo/README.md

```
# Lorenzo ????

1. Lorenzo???????SY???deposit??. (??????????????, ???SY????????deposit)
2. ?5%???, ?????????BANK??.
3. ??`PendleLorenzoSUsd1PlusReward`?`releaseReward`??,
???????PendleLorenzoSUsd1PlusReward???
4. Pendle????claimReward?, ?BANK??SY??, ?????.
```

https://github.com/Lorenzo-Protocol/Pendle-SY-Public/blob/85b59e61e4d22867a872de078708d3b75729d86d/contracts/core/StandardizedYield/implementations/Lorenzo/PendleLorenzoSUsd1PlusReward.sol#L76-L93

```
76      // @dev release the rewards from the SY
77      // @param claimer the claimer address
78      // @param amount the amount of rewards to release
79      function releaseRewards(address claimer, uint256 amount) external onlyOwner {
80          if (claimer == address(0)) {
81              revert InvalidClaimer(claimer);
82          }
83
84          if (amount == 0) {
```

```
85              revert InvalidAmount(amount);
86          }
87
88          rewardState[claimer].pendingAmount  += amount;
89          rewardState[claimer].releasedAmount += amount;
90          rewardState[claimer].lastReleaseTime = uint32(block.timestamp);
91
92          emit ReleaseReward(claimer, amount, uint32(block.timestamp));
93      }
```

Status

✓ Fixed

[WP-I9] PendleLorenzoSUsd1PlusSY uses UUPS which differs from other SYs

Informational

Issue Description

Consider maintaining consistency with other SYs by using TransparentUpgradeableProxy instead of UUPS proxy.

https://github.com/Lorenzo-Protocol/Pendle-SY-Public/blob/85b59e61e4d22867a872de078708d3b75729d86d/contracts/core/StandardizedYield/implementations/Lorenzo/PendleLorenzoSUsd1PlusSY.sol#L9-L126

```
9   contract PendleLorenzoSUsd1PlusSY is UUPSUpgradeable, SYBaseWithRewardsUpg {
    @@ 10,40 @@
41
42      function _authorizeUpgrade(address newImplementation) internal override
    onlyOwner {}
43
    @@ 44,125 @@
126 }
```

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.