Pendle sApe Audit Report

Dev 13, 2022

WATCHPUG

Table of Contents

Summary

This report has been prepared for Pendle sApe 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 sApe**
Codebase	**https://github.com/pendle-finance/pendle-core-v2**
Commit	**c066577d200734cf929f745d4b9e88d6b7ef89ac**
Language	**Solidity**

Audit Summary

Delivery Date	**Dev 13, 2022**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**5**

[WP-H1] A malicious early user/attacker can manipulate the pricePerShare to take an unfair share of future users' deposits

High

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/99f7ece7e51e4119693eec0d1f70f378fb4c21d5/contracts/core/StandardizedYield/implementations/Ape/PendleApeStakingSY.sol#L33-L56

```
33      function _deposit(address, uint256 amountDeposited)
34          internal
35          virtual
36          override
37          returns (uint256 amountSharesOut)
38      {
39          // Respecting APE's deposit invariant & prevent frontrunning on deployment
40          if (amountDeposited < MIN_APE_DEPOSIT) {
41              revert Errors.SYApeDepositAmountTooSmall(amountDeposited);
42          }
43
44          _harvestAndCompound();
45
46          // As SY Base is pulling the tokenIn first, the totalAsset should exclude user's deposit
47          uint256 priorTotalAssetOwned = getTotalAssetOwned() - amountDeposited;
48
49          if (totalSupply() == 0) {
50              amountSharesOut = amountDeposited;
51          } else {
52              // The upcoming calculation can be reduced to amountDeposited.divDown(exchangeRate())
53              // The following calculation is choosen instead to minimize precision error
54              amountSharesOut = (amountDeposited * totalSupply()) / priorTotalAssetOwned;
55          }
56      }
```

A malicious early user can `deposit()` with `MIN_APE_DEPOSIT` of ApeCoin (if they are the first depositor), and withdraw all but a small amount (eg, `199` wei) of the deposit to infalte the pps of the SY.

- `deposit()` `1e18 wei` `apeCoin` and get `1e18 wei` of `PendleApeStakingSY` ;
- `redeem()` `1e18 - 199 wei` of `PendleApeStakingSY`

Then the attacker can send `100e18 - 199` of `apeCoin` tokens and inflate the `exchangeRate()` to `100e18 * 1e18 / 199` .

As a result, the future user who deposits `1e18` will only receive `1e18 * 199 / 100e18 = 1 wei` of SY shares.

They will immediately lose `0.495e18` or half of their deposits if they `redeem()` right after the `deposit()` .

Recommendation

Consider requiring a minimal amount of `PendleApeStakingSY` to be minted for the first minter, and send a portion of the initial mints as a reserve to the DAO so that the pricePerShare can be more resistant to manipulation.

Status

✓ Fixed

[WP-G2] `_redeem()` The `compound` action in `_harvestAndCompound()` may not be necessary

Gas

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/99f7ece7e51e4119693eec0d1f70f378fb4c21d5/contracts/core/StandardizedYield/implementations/Ape/PendleApeStakingSY.sol#L58-L84

```
58      function _redeem(
59          address receiver,
60          address,
61          uint256 amountSharesToRedeem
62      ) internal virtual override returns (uint256 amountTokenOut) {
63          _harvestAndCompound();
64
65          // As SY is burned before calling _redeem(), we should account for priorSupply
66          uint256 priorTotalSupply = totalSupply() + amountSharesToRedeem;
67
68          if (amountSharesToRedeem == priorTotalSupply) {
69              amountTokenOut = getTotalAssetOwned();
70          } else {
71              // The upcoming calculation can be reduced to amountSharesToRedeem.mulDown(exchangeRate())
72              // The following calculation is choosen instead to minimize precision error
73              amountTokenOut = (amountSharesToRedeem * getTotalAssetOwned()) / priorTotalSupply;
74          }
75
76          // There might be case when the contract is holding < 1 APE reward and user is withdrawing everything out of it
77          if (amountTokenOut > _selfBalance(apeCoin)) {
78              IApeStaking(apeStaking).withdrawApeCoin(
79                  amountTokenOut - _selfBalance(apeCoin),
80                  address(this)
81              );
82          }
```

```
83            _transferOut(apeCoin, receiver, amountTokenOut);
84        }
```

At the beginning of the `_redeem()` function, `_harvestAndCompound()` is called to harvest and reinvest (if the balance is greater than the MIN_APE_DEPOSIT).

Later, at L78, `withdrawApeCoin()` will be called to withdraw some.

The reinvestment can be saved if the amount harvested is less than the withdrawal amount.

Recommendation

Consider changing to only `harvest()` at the beginning of `_redeem()`, and check if reinvestment is needed and only do the reinvestment when needed AFTER `apeStaking.withdrawApeCoin()`.

Status

✓ Fixed

[WP-G3] Cache external call results can save gas

Gas

Issue Description

Every call to an external contract costs a decent amount of gas. For optimization of gas usage, external call results should be cached if they are being used for more than one time.

https://github.com/pendle-finance/pendle-core-internal-v2/blob/99f7ece7e51e4119693eec0d1f70f378fb4c21d5/contracts/core/StandardizedYield/implementations/Ape/PendleApeStakingSY.sol#L58-L84

```
58      function _redeem(
59          address receiver,
60          address,
61          uint256 amountSharesToRedeem
62      ) internal virtual override returns (uint256 amountTokenOut) {
63          _harvestAndCompound();
64
65          // As SY is burned before calling _redeem(), we should account for priorSupply
66          uint256 priorTotalSupply = totalSupply() + amountSharesToRedeem;
67
68          if (amountSharesToRedeem == priorTotalSupply) {
69              amountTokenOut = getTotalAssetOwned();
70          } else {
71              // The upcoming calculation can be reduced to amountSharesToRedeem.mulDown(exchangeRate())
72              // The following calculation is choosen instead to minimize precision error
73              amountTokenOut = (amountSharesToRedeem * getTotalAssetOwned()) / priorTotalSupply;
74          }
75
76          // There might be case when the contract is holding < 1 APE reward and user is withdrawing everything out of it
77          if (amountTokenOut > _selfBalance(apeCoin)) {
78              IApeStaking(apeStaking).withdrawApeCoin(
79                  amountTokenOut - _selfBalance(apeCoin),
80                  address(this)
```

```
81                 );
82             }
83             _transferOut(apeCoin, receiver, amountTokenOut);
84         }
```

https://github.com/pendle-finance/pendle-core-internal-v2/blob/99f7ece7e51e4119693eec0d1f70f378fb4c21d5/contracts/core/libraries/TokenHelper.sol#L58-L60

```
58         function _selfBalance(address token) internal view returns (uint256) {
59             return (token == NATIVE) ? address(this).balance :
       IERC20(token).balanceOf(address(this));
60         }
```

`_selfBalance(apeCoin)` can be cached in storage to save the external call.

Status

ⓘ Acknowledged

[WP-G4] Use `_selfBalance(IERC20)` rather than `_selfBalance(address)` can save gas

Gas

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/99f7ece7e51e4119693eec0d1f70f378fb4c21d5/contracts/core/StandardizedYield/implementations/Ape/PendleApeStakingSY.sol#L58-L84

```
58      function _redeem(
59          address receiver,
60          address,
61          uint256 amountSharesToRedeem
62      ) internal virtual override returns (uint256 amountTokenOut) {
63          _harvestAndCompound();
64
65          // As SY is burned before calling _redeem(), we should account for priorSupply
66          uint256 priorTotalSupply = totalSupply() + amountSharesToRedeem;
67
68          if (amountSharesToRedeem == priorTotalSupply) {
69              amountTokenOut = getTotalAssetOwned();
70          } else {
71              // The upcoming calculation can be reduced to amountSharesToRedeem.mulDown(exchangeRate())
72              // The following calculation is choosen instead to minimize precision error
73              amountTokenOut = (amountSharesToRedeem * getTotalAssetOwned()) / priorTotalSupply;
74          }
75
76          // There might be case when the contract is holding < 1 APE reward and user is withdrawing everything out of it
77          if (amountTokenOut > _selfBalance(apeCoin)) {
78              IApeStaking(apeStaking).withdrawApeCoin(
79                  amountTokenOut - _selfBalance(apeCoin),
80                  address(this)
81              );
82          }
```

```
83          _transferOut(apeCoin, receiver, amountTokenOut);
84      }
```

https://github.com/pendle-finance/pendle-core-internal-v2/blob/99f7ece7e51e4119693eec0d1f70f378fb4c21d5/contracts/core/StandardizedYield/implementations/Ape/PendleApeStakingSY.sol#L98-L122

```
98      function getTotalAssetOwned() public view returns (uint256 totalAssetOwned) {
99          (uint256 stakedAmount, ) =
    IApeStaking(apeStaking).addressPosition(address(this));
100         uint256 unclaimedAmount = IApeStaking(apeStaking).pendingRewards(
101             APE_COIN_POOL_ID,
102             address(this),
103             0
104         );
105         uint256 floatingAmount = _selfBalance(apeCoin);
106         totalAssetOwned = stakedAmount + unclaimedAmount + floatingAmount;
107     }
108
109     function _harvestAndCompound() internal {
110         // Claim reward
111         uint256 currentEpochId = _getCurrentEpochId();
112         if (currentEpochId != lastRewardClaimedEpoch) {
113             IApeStaking(apeStaking).claimSelfApeCoin();
114             lastRewardClaimedEpoch = currentEpochId;
115         }
116
117         // Deposit APE
118         uint256 amountAssetToCompound = _selfBalance(apeCoin);
119         if (amountAssetToCompound >= MIN_APE_DEPOSIT) {
120             IApeStaking(apeStaking).depositSelfApeCoin(amountAssetToCompound);
121         }
122     }
```

https://github.com/pendle-finance/pendle-core-internal-v2/blob/99f7ece7e51e4119693eec0d1f70f378fb4c21d5/contracts/core/libraries/TokenHelper.sol#L58-L64

```
58      function _selfBalance(address token) internal view returns (uint256) {
59          return (token == NATIVE) ? address(this).balance :
    IERC20(token).balanceOf(address(this));
60      }
61
62      function _selfBalance(IERC20 token) internal view returns (uint256) {
63          return token.balanceOf(address(this));
64      }
```

Based on the context, we know for a fact that `apeCoin` is not `NATIVE`, therefore, the check of `token == NATIVE` is unnecessary.

Recommendation

Consider using `_selfBalance(IERC20(apeCoin))` to avoid the `token == NATIVE` check.

Status

ⓘ Acknowledged

[WP-G5] `PendleApeStakingSY._deposit()` Avoiding unnecessary external call can save gas

Gas

Issue Description

https://github.com/pendle-finance/pendle-core-internal-v2/blob/99f7ece7e51e4119693eec0d1f70f378fb4c21d5/contracts/core/StandardizedYield/implementations/Ape/PendleApeStakingSY.sol#L33-L56

```
33      function _deposit(address, uint256 amountDeposited)
34          internal
35          virtual
36          override
37          returns (uint256 amountSharesOut)
38      {
39          // Respecting APE's deposit invariant & prevent frontrunning on deployment
40          if (amountDeposited < MIN_APE_DEPOSIT) {
41              revert Errors.SYApeDepositAmountTooSmall(amountDeposited);
42          }
43
44          _harvestAndCompound();
45
46          // As SY Base is pulling the tokenIn first, the totalAsset should exclude
    user's deposit
47          uint256 priorTotalAssetOwned = getTotalAssetOwned() - amountDeposited;
48
49          if (totalSupply() == 0) {
50              amountSharesOut = amountDeposited;
51          } else {
52              // The upcoming calculation can be reduced to
    amountDeposited.divDown(exchangeRate())
53              // The following calculation is choosen instead to minimize precision
    error
54              amountSharesOut = (amountDeposited * totalSupply()) /
    priorTotalAssetOwned;
55          }
56      }
```

https://github.com/pendle-finance/pendle-core-internal-v2/blob/99f7ece7e51e4119693eec0d1f70f378fb4c21d5/contracts/core/StandardizedYield/implementations/Ape/PendleApeStakingSY.sol#L98-L107

```
98      function getTotalAssetOwned() public view returns (uint256 totalAssetOwned) {
99          (uint256 stakedAmount, ) =
    IApeStaking(apeStaking).addressPosition(address(this));
100         uint256 unclaimedAmount = IApeStaking(apeStaking).pendingRewards(
101             APE_COIN_POOL_ID,
102             address(this),
103             0
104         );
105         uint256 floatingAmount = _selfBalance(apeCoin);
106         totalAssetOwned = stakedAmount + unclaimedAmount + floatingAmount;
107     }
```

Only when `totalSupply() != 0` , `getTotalAssetOwned()` is needed, otherwise, , it is not needed and should not be called given the expensive external calls inside it.

Recommendation

Consider changing to:

```
33      function _deposit(address, uint256 amountDeposited)
34          internal
35          virtual
36          override
37          returns (uint256 amountSharesOut)
38      {
39          // Respecting APE's deposit invariant & prevent frontrunning on deployment
40          if (amountDeposited < MIN_APE_DEPOSIT) {
41              revert Errors.SYApeDepositAmountTooSmall(amountDeposited);
42          }
43
44          _harvestAndCompound();
45
46          if (totalSupply() == 0) {
47              amountSharesOut = amountDeposited;
48          } else {
49              // As SY Base is pulling the tokenIn first, the totalAsset should
    exclude user's deposit
```

```
50              uint256 priorTotalAssetOwned = getTotalAssetOwned() - amountDeposited;
51              // The upcoming calculation can be reduced to
    amountDeposited.divDown(exchangeRate())
52              // The following calculation is choosen instead to minimize precision
    error
53              amountSharesOut = (amountDeposited * totalSupply()) /
    priorTotalAssetOwned;
54          }
55      }
```

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.