Pendle sPendle Audit Report

Jan 11, 2026

Table of Contents

Summary

This report has been prepared for sPendle 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	**sPendle**
Codebase	**https://github.com/pendle-finance/pendle-core-v2**
Commit	**e23126085ff1c971024fb535ff1f69f91c752f86**
Language	**Solidity**

Audit Summary

Delivery Date	**Jan 11, 2026**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**5**

[WP-O1] Updated `cooldownDuration` affects in-progress cooldowns which may not meet users' expectations

Issue Description

https://github.com/pendle-finance/pendle-core-v2/blob/3b457f8b75fa57936a31354a05b8bfd18b1d1beb/contracts/LiquidityMining/sPendle/StakedPendle.sol#L41-L49

```
41      function cooldown(uint256 amount) external {
42          require(amount > 0, "sPendle: invalid amount");
43          require(userCooldown[msg.sender].amount == 0, "sPendle: already in
    cooldown");
44
45          userCooldown[msg.sender] = UserCooldown({cooldownStart:
    uint104(block.timestamp), amount: amount.Uint152()});
46          _burn(msg.sender, amount);
47
48          emit CooldownInitiated(msg.sender, amount, block.timestamp);
49      }
```

https://github.com/pendle-finance/pendle-core-v2/blob/3b457f8b75fa57936a31354a05b8bfd18b1d1beb/contracts/LiquidityMining/sPendle/StakedPendle.sol#L61-L79

```
61      function finalizeCooldown()
62          external
63          returns (
64              uint256 /*amount*/
65          )
66      {
67          UserCooldown memory data = userCooldown[msg.sender];
68          require(data.amount > 0, "sPendle: no cooldown");
69
70          uint256 cooldownEnd = data.cooldownStart + cooldownDuration;
71          require(block.timestamp >= cooldownEnd, "sPendle: redeem not ready");
72
73          delete userCooldown[msg.sender];
74          _transferOut(PENDLE, msg.sender, data.amount);
```

```
75
76          emit Unstaked(msg.sender, data.amount, 0);
77
78          return data.amount;
79      }
```

https://github.com/pendle-finance/pendle-core-v2/blob/3b457f8b75fa57936a31354a05b8bfd18b1d1beb/contracts/LiquidityMining/sPendle/StakedPendle.sol#L128-L133

```
128     function setCooldownDurationAndFee(uint24 newDuration, uint64 newInstantUnstakeFeeRate) external onlyOwner {
129         require(newInstantUnstakeFeeRate <= PMath.ONE, "sPendle: invalid fee factor");
130         cooldownDuration = newDuration;
131         instantUnstakeFeeRate = newInstantUnstakeFeeRate;
132         emit CooldownDurationAndFeeUpdated(newDuration, newInstantUnstakeFeeRate);
133     }
```

Recommendation

Consider changing to:

```
41      function cooldown(uint256 amount) external {
42          require(amount > 0, "sPendle: invalid amount");
43          require(userCooldown[msg.sender].amount == 0, "sPendle: already in cooldown");
44
45          uint256 cooldownEnd = data.cooldownStart + cooldownDuration;
46
47          userCooldown[msg.sender] = UserCooldown({cooldownEnd: uint104(cooldownEnd), amount: amount.Uint152()});
48          _burn(msg.sender, amount);
49
50          emit CooldownInitiated(msg.sender, amount, block.timestamp);
51      }
```

Status

ⓘ Acknowledged

[WP-O2] `StakedPendle.initialize()` does not initialize `feeReceiver`, making `StakedPendle.instantUnstake()` unusable until the first call to `StakedPendle.setFeeReceiver()`.

Issue Description

Pendle token does not allow `address(0)` as the `_to` parameter in `transfer(address _to, uint256 _value)`.

https://github.com/pendle-finance/pendle-core-v2/blob/134b10def50c893afd850afa928b9ee0c037eba1/contracts/LiquidityMining/sPendle/StakedPendle.sol

```
@@ 1,8 @@
 9
10  contract StakedPendle is IPStakedPendle, ERC20Upgradeable,
    BoringOwnableUpgradeableV2, TokenHelper {
11      using PMath for uint256;
12
13      address public feeReceiver;
14      address public immutable PENDLE;
15
16      mapping(address user => UserCooldown) public userCooldown;
17
18      uint24 public cooldownDuration;
19      uint64 public instantUnstakeFeeRate; // base 1e18. 5e16 means instant unstake
    will have to pay 5% of the amount.
20
21      constructor(address pendle) {
22          PENDLE = pendle;
23          _disableInitializers();
24      }
25
26      /// @dev reminder to also call setFeeReceiver & setCooldownDurationAndFee
27      function initialize(address _owner) external initializer {
28          __ERC20_init("StakedPendle", "sPENDLE");
29          __BoringOwnableV2_init(_owner);
30      }
31
```

```
@@ 32,74 @@

    function instantUnstake(uint256 amount) external returns (uint256 amountAfterFee, uint256 fee) {
        require(amount > 0, "sPendle: invalid amount");

        fee = PMath.rawDivUp(amount * instantUnstakeFeeRate, PMath.ONE);
        amountAfterFee = amount - fee;

        _burn(msg.sender, amount);
        _transferOut(PENDLE, msg.sender, amountAfterFee);
        _transferOut(PENDLE, feeReceiver, fee);

        emit Unstaked(msg.sender, amountAfterFee, fee);
    }

    // ======== Admin functions ========

    function setFeeReceiver(address newFeeReceiver) external onlyOwner {
        require(newFeeReceiver != address(0), "sPendle: invalid fee receiver");

        feeReceiver = newFeeReceiver;
        emit FeeReceiverUpdated(newFeeReceiver);
    }

    function setCooldownDurationAndFee(uint24 newDuration, uint64 newInstantUnstakeFeeRate) external onlyOwner {
        require(newInstantUnstakeFeeRate <= PMath.ONE, "sPendle: invalid fee factor");
        cooldownDuration = newDuration;
        instantUnstakeFeeRate = newInstantUnstakeFeeRate;
        emit CooldownDurationAndFeeUpdated(newDuration, newInstantUnstakeFeeRate);
    }
}
```

```
@@ 161,166 @@
    function transfer(address dst, uint256 amount) external override returns (bool) {
        _transfer(msg.sender, dst, amount);
        return true;
    }
```

```
362     function _transfer(
363         address src,
364         address dst,
365         uint256 amount
366     ) internal {
367         require(src != address(0), "SENDER_ZERO_ADDR");
368         require(dst != address(0), "RECEIVER_ZERO_ADDR");
369         require(dst != address(this), "SEND_TO_TOKEN_CONTRACT");
370
@@ 371,375 @@
376     }
```

```
113     /**
114      * @dev See {IERC20-transfer}.
115      *
116      * Requirements:
117      *
118      * - `recipient` cannot be the zero address.
119      * - the caller must have a balance of at least `amount`.
120      */
121     function transfer(address recipient, uint256 amount) public virtual override
    returns (bool) {
122         _transfer(_msgSender(), recipient, amount);
123         return true;
124     }
```

```
200     /**
201      * @dev Moves tokens `amount` from `sender` to `recipient`.
202      *
203      * This is internal function is equivalent to {transfer}, and can be used to
204      * e.g. implement automatic token fees, slashing mechanisms, etc.
205      *
206      * Emits a {Transfer} event.
207      *
208      * Requirements:
209      *
210      * - `sender` cannot be the zero address.
211      * - `recipient` cannot be the zero address.
212      * - `sender` must have a balance of at least `amount`.
```

```
213       */
214      function _transfer(address sender, address recipient, uint256 amount) internal
     virtual {
215          require(sender != address(0), "ERC20: transfer from the zero address");
216          require(recipient != address(0), "ERC20: transfer to the zero address");
217
     @@ 218,222 @@
223      }
```

Status

ⓘ Acknowledged

[WP-N3] Unnecessary `payable` modifier on `StakedPendle.multicall()`

Issue Description

The `StakedPendle` contract has no native token functionality and no other payable functions. Therefore, the `payable` modifier on `StakedPendle.multicall()` is unnecessary.

More broadly, having a payable multicall can easily lead to accidental multiple spending of `msg.value` . Avoiding the payable modifier can reduce the mental overhead when using multicall.

https://github.com/pendle-finance/pendle-core-v2/blob/3b457f8b75fa57936a31354a05b8bfd18b1d1beb/contracts/LiquidityMining/sPendle/StakedPendle.sol

```
@@ 1,8 @@

contract StakedPendle is IPStakedPendle, ERC20Upgradeable,
BoringOwnableUpgradeableV2, TokenHelper {
@@ 11,92 @@

    // ======== Misc helper functions =======

    function multicall(Call3[] calldata calls) external payable returns (Result[]
memory res) {
        uint256 length = calls.length;
        res = new Result[](length);
        for (uint256 i = 0; i < length; i++) {
            (bool success, bytes memory result) =
_delegateToSelf(calls[i].callData, calls[i].allowFailure);
            res[i] = Result(success, result);
        }
    }

    function _delegateToSelf(bytes memory data, bool allowFailure)
        internal
        returns (bool success, bytes memory result)
    {
```

```
109         (success, result) = address(this).delegatecall(data);
110
111         if (!success && !allowFailure) {
112             assembly {
113                 // We use Yul's revert() to bubble up errors from the target
    contract.
114                 revert(add(32, result), mload(result))
115             }
116         }
117     }
118
    @@ 119,133 @@
134 }
```

Status

✓ Fixed

[WP-N4] Some error messages are no longer needed

Issue Description

contracts/core/libraries/Errors.sol

```
121     // Liquidity Mining
122     error VCInvalidCap(uint256 cap);
123     error VCInactivePool(address pool);
124     error VCPoolAlreadyActive(address pool);
125     error VCZeroVePendle(address user);
126     error VCExceededMaxWeight(uint256 totalWeight, uint256 maxWeight);
127     error VCEpochNotFinalized(uint256 wTime);
128     error VCPoolAlreadyAddAndRemoved(address pool);
```

```
140 error GCNotPendleMarket(address caller);
```

https://github.com/pendle-finance/pendle-core-v2/blob/3b457f8b75fa57936a31354a05b8bfd18b1d1beb/contracts/core/libraries/Errors.sol#L147-L157

```
147 error FDTotalAmountFundedNotMatch(uint256 actualTotalAmount, uint256
    expectedTotalAmount);
148 error FDEpochLengthMismatch();
149 error FDInvalidPool(address pool);
150 error FDPoolAlreadyExists(address pool);
151 error FDInvalidNewFinishedEpoch(uint256 oldFinishedEpoch, uint256
    newFinishedEpoch);
152 error FDInvalidStartEpoch(uint256 startEpoch);
153 error FDInvalidWTimeFund(uint256 lastFunded, uint256 wTime);
154 error FDFutureFunding(uint256 lastFunded, uint256 currentWTime);
155
156 error BDInvalidEpoch(uint256 epoch, uint256 startTime);
157
```

Status

ⓘ Acknowledged

[WP-N5] Unused `PMath` , `SafeERC20` in `PendleGauge.sol`

Issue Description

`PendleGauge` does not use any code from `PMath` and `SafeERC20` .

https://github.com/pendle-finance/pendle-core-v2/blob/3b457f8b75fa57936a31354a05b8bfd18b1d1beb/contracts/core/Market/PendleGauge.sol

```
1  // SPDX-License-Identifier: GPL-3.0-or-later
2  pragma solidity ^0.8.0;
3
4  import "../../interfaces/IPGauge.sol";
5  import "../../interfaces/IPGaugeController.sol";
6  import "../../interfaces/IStandardizedYield.sol";
7
8  import "../RewardManager/RewardManager.sol";
9
10 abstract contract PendleGauge is RewardManager, IPGauge {
11     using PMath for uint256;
12     using SafeERC20 for IERC20;
13     using ArrayLib for address[];
14
15     address private immutable SY;
16     address internal immutable PENDLE;
17     address internal immutable gaugeController;
18
19     constructor(address _SY, address _gaugeController) {
20         SY = _SY;
21         gaugeController = _gaugeController;
22         PENDLE = IPGaugeController(gaugeController).pendle();
23     }
24
25     function _redeemRewards(address user) internal virtual returns (uint256[]
   memory rewardsOut) {
26         _updateAndDistributeRewards(user);
27         rewardsOut = _doTransferOutRewards(user, user);
28         emit RedeemRewards(user, rewardsOut);
29     }
30
31     function _redeemExternalReward() internal virtual override {
32         IStandardizedYield(SY).claimRewards(address(this));
```

```
33          IPGaugeController(gaugeController).redeemMarketReward();
34      }
35
36      function _getRewardTokens() internal view virtual override returns (address[]
    memory) {
37          address[] memory SYRewards = IStandardizedYield(SY).getRewardTokens();
38          if (SYRewards.contains(PENDLE)) return SYRewards;
39          return SYRewards.append(PENDLE);
40      }
41
42      function _beforeTokenTransfer(address from, address to, uint256) internal
    virtual {
43          _updateAndDistributeRewardsForTwo(from, to);
44      }
45  }
```

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.