PenPie Audit Report

Jun 5, 2023

WATCHPUG

Table of Contents

Summary

This report has been prepared for PenPie 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	**PenPie**
Codebase	**https://github.com/magpiexyz/pendleMagpie**
Commit	**0470a8c5ad5021c870d22bb5dbb348f69120a109**
Language	**Solidity**

Audit Summary

Delivery Date	**Jun 5, 2023**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**7**

[WP-H1] `harvestVePendleReward()` Lack of methods to claim vePendle rewards

High

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/1e23ec6148f9fb3a081d3efd89f87a04cbf687cb/contracts/pendle/PendleStaking.sol#L330-L373

```
330  function registerPool(
331      address _market,
332      uint256 _allocPoints,
333      string memory name,
334      string memory symbol
335  ) external onlyOwner {
336      if (pools[_market].isActive != false) {
337          revert PoolOccupied();
338      }
339
340      IERC20 newToken = IERC20(
341          ERC20FactoryLib.createReceipt(_market, masterPenpie, name, symbol)
342      );
343
344      address rewarder = IMasterPenpie(masterPenpie).createRewarder(
345          address(newToken),
346          address(PENDLE)
347      );
348
349      IPendleMarketDepositHelper(marketDepositHelper).setPoolInfo(
350          _market,
351          rewarder,
352          true
353      );
354
355      IMasterPenpie(masterPenpie).add(
356          _allocPoints,
357          address(_market),
358          address(newToken),
359          address(rewarder)
360      );
```

```
361
362     pools[_market] = Pool({
363         isActive: true,
364         market: _market,
365         receiptToken: address(newToken),
366         rewarder: address(rewarder),
367         helper: marketDepositHelper,
368         lastHarvestTime: block.timestamp
369     });
370     poolTokenList.push(_market);
371
372     emit PoolAdded(_market, address(rewarder), address(newToken));
373 }
```

https://github.com/magpiexyz/pendleMagpie/blob/1e23ec6148f9fb3a081d3efd89f87a04cbf687cb/contracts/pendle/PendleStaking.sol#L289-L317

```
289 function harvestVePendleReward() external {
290     if (this.totalUnclaimedETH() == 0)
291         revert NoVePendleReward();
292
293     if ((protocolFee != 0 && feeCollector == address(0)) || bribeManagerEOA ==
    address(0))
294         revert InvalidFeeDestination();
295
296     uint256 length = poolTokenList.length;
297     address[] memory _pools = new address[](length);
298
299     for (uint256 i; i < length; i++) {
300         _pools[i] = poolTokenList[i];
301     }
302
303     (uint256 totalAmountOut, ) = distributorETH.claimProtocol(address(this),
    _pools);
304     // for protocol
305     uint256 fee = (totalAmountOut * protocolFee) / DENOMINATOR;
306     IERC20(WETH).safeTransfer(feeCollector, fee);
307
308     // for caller
309     uint256 callerFeeAmount = (totalAmountOut * vePendleHarvestCallerFee) /
    DENOMINATOR;
```

```
310         IERC20(WETH).safeTransfer(msg.sender, callerFeeAmount);
311
312         uint256 left = totalAmountOut - fee - callerFeeAmount;
313         IERC20(WETH).safeTransfer(bribeManagerEOA, left);
314
315         emit VePendleHarvested(totalAmountOut, fee, callerFeeAmount, left);
316     }
```

The `_pools` parameter in the `harvestVePendleReward()` function does not include `vePendle`. Consequently, a significant portion of the rewards accredited to the `vePendle` held by the `PendleStaking` contract can not be claimed.

Furthermore, we have been unable to locate any other location where the `vePendle` rewards can be claimed.

Recommendation

Consider adding the `vePendle` address to the `_pools` list to claim the vePendle rewards.

Status

✓ Fixed

[WP-H2] The current setup doesn't work on Arbitrum.

High

Issue Description

Based on the deploy script, it appears that you intend to deploy the exact same code for `mPendleConvertor` and `PendleStaking` on Arbitrum.

However, this will not work as Pendle does not allow locking PENDLE to vePendle on Arbitrum.

The expected setup should be as follows:

1. If you plan to enable users to convert Pendle to mPendle on Arbitrum, the `mPendleConvertor` should allow the admin to withdraw the Pendle tokens and bridge them to the mainchain, then lock them to vePendle.
2. The `PendleStaking` contract on the sidechain should synchronize the vePendle balance from the mainchain and hold the LP tokens on behalf of the users for boosting and rewards.

https://github.com/magpiexyz/pendleMagpie/blob/46f24334e7c075813767a24c30512a3a1e62e292/contracts/pendle/PendleStaking.sol#L243-L260

```
243  function convertPendle(
244      uint256 _amount
245  ) public whenNotPaused returns (uint256) {
246      uint256 preVePendleAmount = this.accumulatedVePendle();
247      if (_amount > 0) {
248          IERC20(PENDLE).safeApprove(address(vePendle), _amount);
249          uint128 unlockTime = _getIncreaseLockTime();
250          IPVotingEscrowMainchain(vePendle).increaseLockPosition(
251              uint128(_amount),
252              unlockTime
253          );
254      }
255      uint256 mintedVePendleAmount = this.accumulatedVePendle() -
256          preVePendleAmount;
257      emit PendleLocked(_amount, lockPeriod, mintedVePendleAmount);
258
259      return mintedVePendleAmount;
```

```
260   }
```

Status

✓ **Fixed**

[WP-M3] Griefing attack by calling `IPendleMarket.redeemRewards()` to claim the reward for `PendleStaking`, resulting in the rewardAmount cannot be correctly assigned to the pools

Medium

Issue Description

`PendleMarket.redeemRewards()` allows any `msg.sender`, not just the user, to claim rewards on behalf of the user.

However, `PendleStaking#_harvestMarketRewards()` relies on the delta amount of `rewardTokens` balances to track the rewards.

If an attack were to call `IPendleMarket.redeemRewards()` for `PendleStaking`, it could prevent `_harvestMarketRewards()` from accurately tracking the reward amounts.

https://github.com/magpiexyz/pendleMagpie/blob/5ee6019761b7055e5ea5f8df70a18fa452ca8dc1/contracts/pendle/PendleStaking.sol#L565-L587

```
565  function _harvestMarketRewards(address _market, bool _force) internal {
566      Pool storage poolInfo = pools[_market];
567      if (!_force && (block.timestamp - poolInfo.lastHarvestTime) < harvestTimeGap)
568          return;
569
570      address[] memory bonusTokens = IPendleMarket(_market).getRewardTokens();
571      uint256[] memory beforeBalances = _rewardBeforeBalances(bonusTokens);
572      IPendleMarket(_market).redeemRewards(address(this));
573
574      for (uint256 i; i < bonusTokens.length; i++) {
575          uint256 bonusBalanceDiff = IERC20(bonusTokens[i]).balanceOf(
576              address(this)
577          ) - beforeBalances[i];
578          if (bonusBalanceDiff > 0) {
579              _sendRewards(
580                  _market,
581                  bonusTokens[i],
582                  poolInfo.rewarder,
```

```
583                     bonusBalanceDiff
584                 );
585             }
586         }
587     }
```

Recommendation

Consider monitoring the source of rewards claimed on behalf of `PendleMarket` off-chain and allocating the claimed rewards based on the monitoring data.

Status

✓ Fixed

[WP-M4] `MasterPenpie._onlyWhiteListed` is improperly implemented, resulting in a malfunction of `updatePoolsAlloc()`.

Medium

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/5d85c87f55f5d19ab87330255f59e5735a43919b/contracts/rewards/MasterPenpie.sol#L145-L154

```
145  modifier _onlyWhiteListed() {
146      if (AllocationManagers[msg.sender])
147          return;
148      if (PoolManagers[msg.sender])
149          return;
150      if (msg.sender == owner())
151          return;
152      revert OnlyWhiteListedAllocaUpdator();
153      _;
154  }
```

https://github.com/magpiexyz/pendleMagpie/blob/1e23ec6148f9fb3a081d3efd89f87a04cbf687cb/contracts/rewards/MasterPenpie.sol#L739-L754

```
739  function updatePoolsAlloc(address[] calldata _stakingTokens, uint256[] calldata
     _allocPoints) external _onlyWhiteListed() {
740      massUpdatePools();
741
742      if (_stakingTokens.length != _allocPoints.length)
743          revert LengthMismatch();
744
745      for (uint256 i = 0; i < _stakingTokens.length; i++) {
746          uint256 oldAllocPoint = tokenToPoolInfo[_stakingTokens[i]].allocPoint;
747
748          totalAllocPoint = totalAllocPoint - oldAllocPoint + _allocPoints[i];
749
750          tokenToPoolInfo[_stakingTokens[i]].allocPoint = _allocPoints[i];
751
752          emit UpdatePoolAlloc(_stakingTokens[i], oldAllocPoint, _allocPoints[i]);
```

```
753         }
754     }
```

A modifier in Solidity is not a function. When it is used with the `return` keyword, it returns the consumer function of the modifier.

Recommendation

```
145     modifier _onlyWhiteListed() {
146         if (AllocationManagers[msg.sender] || PoolManagers[msg.sender] || msg.sender
        == owner()) {
147             _;
148         } else {
149             revert OnlyWhiteListedAllocaUpdator();
150         }
151     }
```

Status

✓ Fixed

[WP-L6] `PendleStaking#_perPoolReward()` was never called

Low

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/5ee6019761b7055e5ea5f8df70a18fa452ca8dc1/contracts/pendle/PendleStaking.sol#L553-L563

```
553  function _perPoolReward(
554      address _pool
555  ) internal nonReentrant returns (uint256) {
556      address[] memory poolAddress = new address[](1);
557      poolAddress[0] = address(_pool);
558      uint256 wethPreBalance = IWETH(WETH).balanceOf(address(this));
559      distributorETH.claimProtocol(address(this), poolAddress);
560      uint256 wethReward = IWETH(WETH).balanceOf(address(this)) -
561          wethPreBalance;
562      return wethReward;
563  }
```

The internal function `_perPoolReward()` in `PendleStaking` is never called anywhere in the contract.

Recommendation

Consider removing the redundant function or put it in use.

Status

✓ **Fixed**

[WP-N7] Dev related codes to be removed

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/1e23ec6148f9fb3a081d3efd89f87a04cbf687cb/contracts/bribeMarket/PenpieBribePool.sol#L8-L10

```
8
9  import "hardhat/console.sol";
10
```

Status

✓ Fixed

[WP-I8] Allowing a 0 amount in `convertPendle()` is unnecessary.

Informational

Issue Description

If the case where `_amount` is 0 is the same as when it is greater than 0, returning early can save gas.

If it is different, specific code is needed.

https://github.com/magpiexyz/pendleMagpie/blob/1e23ec6148f9fb3a081d3efd89f87a04cbf687cb/contracts/pendle/PendleStaking.sol#L243-L260

```
243  function convertPendle(
244      uint256 _amount
245  ) public whenNotPaused returns (uint256) {
246      uint256 preVePendleAmount = this.accumulatedVePendle();
247      if (_amount > 0) {
248          IERC20(PENDLE).safeApprove(address(vePendle), _amount);
249          uint128 unlockTime = _getIncreaseLockTime();
250          IPVotingEscrowMainchain(vePendle).increaseLockPosition(
251              uint128(_amount),
252              unlockTime
253          );
254      }
255      uint256 mintedVePendleAmount = this.accumulatedVePendle() -
256          preVePendleAmount;
257      emit PendleLocked(_amount, lockPeriod, mintedVePendleAmount);
258
259      return mintedVePendleAmount;
260  }
```

Recommendation

Consider requiring `_amount > 0` .

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.