Penpie Bribe Market Audit Report

Jun 28, 2023

Table of Contents

Summary

This report has been prepared for Penpie Bribe Market 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 Bribe Market**
Codebase	**https://github.com/magpiexyz/pendleMagpie**
Commit	**4c7c35768458a51bcf9d314298fba7a79c5c682a**
Language	**Solidity**

Audit Summary

Delivery Date	**Jun 28, 2023**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**17**

[WP-H1] `removePool()` will cause users who allocated their votes to the pool to be unable to unlock their PNP, resulting in their funds being frozen in the contract.

High

Issue Description

For a user who has allocated a certain weight to a pool that was once active but is now inactive, they must remove the weight from this pool in order to unlock their PNP.

However, the current implementation does not allow the user to remove weight from an inactive pool. As a result, the user's PNP will be frozen in the contract.

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerBaseUpg.sol#L184-L220

```
function _updateVoteAndCheck(address _user, UserVote[] memory _userVotes) internal {
    uint256 length = _userVotes.length;
    int256 totalUserVote;

    for (uint256 i; i < length; i++) {
        Pool storage pool = poolInfos[_userVotes[i].pid];
        if (!pool.isActive) revert PoolNotActive();

        int256 weight = _userVotes[i].weight;
        totalUserVote += weight;

        if (weight != 0) {
            if (weight > 0) {
                uint256 absVal = uint256(weight);
                pool.totalVoteInVlPenpie += absVal;
                userVotedForPoolInVlPenpie[_user][pool.market] += absVal;
            } else {
                uint256 absVal = uint256(-weight);
                pool.totalVoteInVlPenpie -= absVal;
                userVotedForPoolInVlPenpie[_user][pool.market] -= absVal;
```

```
            }
        }

        _afterVoteUpdate(_user, pool.market, _userVotes[i].pid, weight);
    }

    // update user's total vote and all vlPNP vote
    if (totalUserVote > 0) {
        userTotalVotedInVlPenpie[_user] += uint256(totalUserVote);
        totalVlPenpieInVote += uint256(totalUserVote);
    } else {
        userTotalVotedInVlPenpie[_user] -= uint256(-totalUserVote);
        totalVlPenpieInVote -= uint256(-totalUserVote);
    }

    if (userTotalVotedInVlPenpie[_user] > getUserVotable(_user)) revert NotEnoughVote();
}
```

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerBaseUpg.sol#L228-L236

```
function _recVoteBribe(address _user, UserVote[] memory _userVotes) internal {
    uint256 length = _userVotes.length;

    for (uint256 i; i < length; i++) {
        Pool storage pool = poolInfos[_userVotes[i].pid];
        if (!pool.isActive) revert PoolNotActive();
        int256 weight = _userVotes[i].weight;
    }
}
```

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/VLPenPie.sol#L274-L307

```
function startUnlock(uint256 _amountToCoolDown) external override whenNotPaused nonReentrant {
    if (_amountToCoolDown > getUserTotalLocked(msg.sender))
```

```
        revert NotEnoughLockedPenpie();

    uint256 totalLockAfterStartUnlock = getUserTotalLocked(msg.sender) - _amountToCoolDown;
    if (address(pendleVoteManager) != address(0) &&
        totalLockAfterStartUnlock < IPendleVoteManager(pendleVoteManager).userTotalVotedInVlPenpie(msg.sender))
        revert NotEnoughLockedPenpie();

    address[] memory lps = new address[](1);
    address[][] memory vlPenpieRewards = new address[][](1);
    lps[0] = address(this);
    IMasterPenpie(masterPenpie).multiclaimFor(lps, vlPenpieRewards, msg.sender);

    uint256 _slotIndex = getNextAvailableUnlockSlot(msg.sender);
    totalAmountInCoolDown += _amountToCoolDown;

    if (_slotIndex < getUserUnlockSlotLength(msg.sender)) {
        userUnlockings[msg.sender][_slotIndex] = UserUnlocking({
            startTime: block.timestamp,
            endTime: block.timestamp + coolDownInSecs,
            amountInCoolDown: _amountToCoolDown
        });
    } else {
        userUnlockings[msg.sender].push(
            UserUnlocking({
                startTime: block.timestamp,
                endTime: block.timestamp + coolDownInSecs,
                amountInCoolDown: _amountToCoolDown
            })
        );
    }
    emit UnlockStarts(msg.sender, block.timestamp, _amountToCoolDown);
}
```

Recommendation

Change to:

```
function _updateVoteAndCheck(address _user, UserVote[] memory _userVotes) internal {
    uint256 length = _userVotes.length;
    int256 totalUserVote;

    for (uint256 i; i < length; i++) {
        Pool storage pool = poolInfos[_userVotes[i].pid];

        int256 weight = _userVotes[i].weight;
        totalUserVote += weight;

        if (weight != 0) {
            if (weight > 0) {
                if (!pool.isActive) revert PoolNotActive();
                uint256 absVal = uint256(weight);
                pool.totalVoteInVlPenpie += absVal;
                userVotedForPoolInVlPenpie[_user][pool.market] += absVal;
            } else {
                uint256 absVal = uint256(-weight);
                pool.totalVoteInVlPenpie -= absVal;
                userVotedForPoolInVlPenpie[_user][pool.market] -= absVal;
            }
        }

        _afterVoteUpdate(_user, pool.market, _userVotes[i].pid, weight);
    }

    // update user's total vote and all vlPNP vote
    if (totalUserVote > 0) {
        userTotalVotedInVlPenpie[_user] += uint256(totalUserVote);
        totalVlPenpieInVote += uint256(totalUserVote);
    } else {
        userTotalVotedInVlPenpie[_user] -= uint256(-totalUserVote);
        totalVlPenpieInVote -= uint256(-totalUserVote);
    }

    if (userTotalVotedInVlPenpie[_user] > getUserVotable(_user)) revert NotEnoughVote();
}
```

Status

✓ Fixed

[WP-H2] Votes for a `pool` that has already been removed from `PendleVotingController.allActivePools` must be excluded to prevent the entire voting process from being blocked by the `pool` .

High

Issue Description

If `PendleVotingController.vote()` detects a non-active `pool` with `weight != 0` , it will `revert Errors.VCInactivePool(pool)` .

However, the current implementation cannot ensure that the votes for a once active pool can be excluded or reset to 0. As a result, the removal of an active pool on Pendle's side can block the voting of the entire system, effectively paralyzing the system.

https://github.com/pendle-finance/pendle-core-v2-public/blob/310bcc9e419b2122eaf65fd283f809023ceddae6/contracts/LiquidityMining/VotingController/PendleVotingControllerUpg.sol#L90

```
function vote(address[] calldata pools, uint64[] calldata weights) external {
    address user = msg.sender;

    if (pools.length != weights.length) revert Errors.ArrayLengthMismatch();
    if (user != owner && vePendle.balanceOf(user) == 0) revert Errors.VCZeroVePendle(user);

    LockedPosition memory userPosition = _getUserVePendlePosition(user);

    for (uint256 i = 0; i < pools.length; ++i) {
        if (_isPoolActive(pools[i])) applyPoolSlopeChanges(pools[i]);
        VeBalance memory newVote = _modifyVoteWeight(user, pools[i], userPosition, weights[i]);
        emit Vote(user, pools[i], weights[i], newVote);
    }

    uint256 totalVotedWeight = userData[user].totalVotedWeight;
    if (totalVotedWeight > VeBalanceLib.USER_VOTE_MAX_WEIGHT)
```

```
        revert Errors.VCExceededMaxWeight(totalVotedWeight, VeBalanceLib.USER_VOTE_MAX_WEIGHT);
}
```

https://github.com/pendle-finance/pendle-core-v2-public/blob/310bcc9e419b2122eaf65fd283f809023ceddae6/contracts/LiquidityMining/VotingController/VotingControllerStorageUpg.sol#L244

```
function _modifyVoteWeight(
    address user,
    address pool,
    LockedPosition memory userPosition,
    uint64 weight
) internal returns (VeBalance memory newVote) {
    UserData storage uData = userData[user];
    PoolData storage pData = poolData[pool];

    VeBalance memory oldVote = uData.voteForPools[pool].vote;

    // REMOVE OLD VOTE
    if (oldVote.bias != 0) {
        if (_isPoolActive(pool) && _isVoteActive(oldVote)) {
            pData.totalVote = pData.totalVote.sub(oldVote);
            pData.slopeChanges[oldVote.getExpiry()] -= oldVote.slope;
        }
        uData.totalVotedWeight -= uData.voteForPools[pool].weight;
        delete uData.voteForPools[pool];
    }

    // ADD NEW VOTE
    if (weight != 0) {
        if (!_isPoolActive(pool)) revert Errors.VCInactivePool(pool);

        newVote = userPosition.convertToVeBalance(weight);

        pData.totalVote = pData.totalVote.add(newVote);
        pData.slopeChanges[newVote.getExpiry()] += newVote.slope;

        uData.voteForPools[pool] = UserPoolData(weight, newVote);
        uData.totalVotedWeight += weight;
    }
```

```

    emit PoolVoteChange(pool, pData.totalVote);
}
```

https://github.com/pendle-finance/pendle-core-v2-public/blob/310bcc9e419b2122eaf65fd283f809023ceddae6/contracts/LiquidityMining/VotingController/VotingControllerStorageUpg.sol#L267

```
function _isPoolActive(address pool) internal view returns (bool) {
    return allActivePools.contains(pool);
}
```

While the manager can remove a pool and mark it as inactive, `PendleVoteManagerMainChain.castVotes()` does not properly process inactive pools.

For an inactive `pool` that `pendleStaking` has previously voted in `PendleVotingController`, it needs to vote `0` weight in order to release the voting power occupied by the inactive `pool`.

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L107-L136

```
function castVotes() override public payable
{
    lastCastTime = block.timestamp;
    uint256 length = poolInfos.length;

    address[] memory _pools = new address[](length);
    uint64[] memory votes = new uint64[](length);

    for (uint256 i; i < length; i++) {
        Pool storage pool = poolInfos[i];
        _pools[i] = pool.market;

        uint256 currentVote = getVoteForMarket(pool.market);
        uint256 targetVoteInVlPenpie = pool.totalVoteInVlPenpie;
        uint256 targetVote = 0;

```

```
        if (totalVlPenpieInVote != 0) {
            targetVote =(targetVoteInVlPenpie * totalVotes()) / totalVlPenpieInVote;
        }

        if (targetVote >= currentVote)
            votes[i] = _getVoteInPercentage(int256(targetVote - currentVote), currentVote);
        else
            votes[i] = _getVoteInPercentage(int256(targetVote) - int256(currentVote), currentVote);

    }

    IPendleStaking(pendleStaking).vote(_pools, votes);
    emit VoteCasted(msg.sender, lastCastTime);
}
```

Recommendation

Consider changing to:

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L107-L141

```
function castVotes() override public payable
{
    lastCastTime = block.timestamp;
    uint256 length = poolInfos.length;

    address[] memory _pools = new address[](length);
    uint64[] memory votes = new uint64[](length);

    for (uint256 i; i < length; i++) {
        Pool storage pool = poolInfos[i];
        _pools[i] = pool.market;

        if (!pool.isActive) {
            // keep `votes[i]` as 0
```

```
            continue;
        }

        uint256 currentVote = getVoteForMarket(pool.market);
        uint256 targetVoteInVlPenpie = pool.totalVoteInVlPenpie;
        uint256 targetVote = 0;

        if (totalVlPenpieInVote != 0) {
            targetVote =(targetVoteInVlPenpie * totalVotes()) / totalVlPenpieInVote;
        }

        if (targetVote >= currentVote)
            votes[i] = _getVoteInPercentage(int256(targetVote - currentVote), currentVote);
        else
            votes[i] = _getVoteInPercentage(int256(targetVote) - int256(currentVote), currentVote);

    }

    IPendleStaking(pendleStaking).vote(_pools, votes);
    emit VoteCasted(msg.sender, lastCastTime);
}
```

Status

✓ **Fixed**

[WP-H3] Wrong implementation of `castVotes()` causes the voting function to not work properly.

High

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L107-L167

```
function castVotes() override public payable
{
    lastCastTime = block.timestamp;
    uint256 length = poolInfos.length;

    address[] memory _pools = new address[](length);
    uint64[] memory votes = new uint64[](length);

    for (uint256 i; i < length; i++) {
        Pool storage pool = poolInfos[i];
        _pools[i] = pool.market;

        uint256 currentVote = getVoteForMarket(pool.market);
        uint256 targetVoteInVlPenpie = pool.totalVoteInVlPenpie;
        uint256 targetVote = 0;

        if (totalVlPenpieInVote != 0) {
            targetVote =(targetVoteInVlPenpie * totalVotes()) / totalVlPenpieInVote;
        }

        if (targetVote >= currentVote)
            votes[i] = _getVoteInPercentage(int256(targetVote - currentVote), currentVote);
        else
            votes[i] = _getVoteInPercentage(int256(targetVote) - int256(currentVote), currentVote);

    }
```

```

    IPendleStaking(pendleStaking).vote(_pools, votes);
    emit VoteCasted(msg.sender, lastCastTime);
}

/* ============ Internal Functions ============ */

function _getVoteInPercentage(int256 _vote, uint256 _currentVote) internal view returns(uint64) {
    uint256 votePerc;
    uint64 pendleVote;
    uint256 exactVoteCount;
    if(_vote >= 0) {
        exactVoteCount = uint256(_vote) + _currentVote;
        votePerc = exactVoteCount * 100 / totalVotes();
        pendleVote = uint64(_getExactPercentage(votePerc));
        } else {
            int256 _votePos = Math.neg(_vote);
            exactVoteCount = _currentVote - uint256(_votePos);
            if(exactVoteCount == 0)
                pendleVote = 0;
            else {
                votePerc = exactVoteCount * 100 / totalVotes();
                pendleVote = uint64(_getExactPercentage(votePerc));
            }
        }
    return pendleVote;
}

function _getExactPercentage(uint256 _exactVotes) internal pure returns(uint256 _exactPercentage){
    uint256 remainder  = _exactVotes % 10;
    if(remainder <=5)
        _exactPercentage = _exactVotes * 1e16;
    else
        _exactPercentage = (_exactVotes + 1) * 1e16;
}
```

`_getExactPercentage` accepts `votePerc` as a parameter.

`votePerc = targetVoteInVlPenpie*100/totalVlPenpieInVote` is the voting percentage.

If the `votePerc % 10 > 5` , then add 1%.

Assume that there are three voting pools with voting percentages of `39(%)` , `39(%)` , and `22(%)` , respectively. `_getExactPercentage` will change them to `40(%)` , `40(%)` , and `22(%)` , making their sum exceed `100(%)` . As a sequence, the voting in pendle will revert.

Recommendation

Consider changing to:

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L107-L123

```
    function castVotes() override public payable
    {
        lastCastTime = block.timestamp;
        uint256 length = poolInfos.length;

        address[] memory _pools = new address[](length);
        uint64[] memory votes = new uint64[](length);

        for (uint256 i; i < length; i++) {
            Pool storage pool = poolInfos[i];
            _pools[i] = pool.market;
            votes[i] = SafeCast.toUint64(pool.totalVoteInVlPenpie * PENDLE_USER_VOTE_MAX_WEIGHT / totalVlPenpieInVote);
        }

        IPendleStaking(pendleStaking).vote(_pools, votes);
        emit VoteCasted(msg.sender, lastCastTime);
    }
```

Note: constant `PENDLE_USER_VOTE_MAX_WEIGHT` should be `1e18` .

Status

✓ Fixed

[WP-M4] `_recVoteBribe()` incomplete implementation.

Medium

Issue Description

`_recVoteBribe()` is a no-op implementation.

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerBaseUpg.sol#L228-L236

```
228 function _recVoteBribe(address _user, UserVote[] memory _userVotes) internal {
229     uint256 length = _userVotes.length;
230
231     for (uint256 i; i < length; i++) {
232         Pool storage pool = poolInfos[_userVotes[i].pid];
233         if (!pool.isActive) revert PoolNotActive();
234         int256 weight = _userVotes[i].weight;
235     }
236 }
```

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerSideChain.sol#L120-L123

```
120 function _nonblockingLzReceive(uint16 _srcChainId, bytes memory _srcAddress,
    uint64 _nonce, bytes memory _payload) internal override {
121     (address user, UserVote[] memory userVotes) = decodeVote(_payload);
122     _recVoteBribe(user, userVotes);
123 }
```

As a reference, this is the implementation of `_recVoteBribe()` in the `main` branch:

https://github.com/magpiexyz/pendleMagpie/blob/5fe72621f4bc9ebe634edb5aa08bd00499c2e56e/contracts/bribeMarket/PendleVoteManagerBaseUpg.sol#L261-L276

```
function _recVoteBribe(address _user, UserVote[] memory _userVotes) internal {
    uint256 length = _userVotes.length;

    for (uint256 i; i < length; i++) {
        Pool storage pool = poolInfos[_userVotes[i].pid];
        if (!pool.isActive) revert PoolNotActive();
        int256 weight = _userVotes[i].weight;
        if (weight != 0) {
            if (weight > 0) {
                IPenpieBribePool(pool.bribe).voteFor(_user, uint256(weight));
            } else {
                IPenpieBribePool(pool.bribe).unvoteFor(_user, uint256(-weight));
            }
        }
    }
}
```

Status

✓ **Fixed**

[WP-M5] `unCollectedFee` in the native token cannot be manually claimed.

Medium

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PenpieBribeManager.sol#L327-L336

```
327 function manualClaimFees(address _token) external onlyOwner {
328     uint256 balance = IERC20(_token).balanceOf(address(this));
329     if (feeCollector != address(0)) {
330         unCollectedFee[_token] = 0;
331         if (_token == NATIVE)
332             feeCollector.transfer(address(this).balance);
333         else
334             IERC20(_token).safeTransfer(feeCollector, balance);
335     }
336 }
```

L328 will revert when `_token == NATIVE` , thus `unCollectedFee` in the native token cannot be manually claimed.

Recommendation

Change to:

```
function manualClaimFees(address _token) external onlyOwner {
    if (feeCollector != address(0)) {
        unCollectedFee[_token] = 0;
        if (_token == NATIVE) {
            feeCollector.transfer(address(this).balance);
        } else {
            uint256 balance = IERC20(_token).balanceOf(address(this));
            IERC20(_token).safeTransfer(feeCollector, balance);
        }
```

```
        }
    }
```

Status

✓ Fixed

[WP-M6] `addBribeNative()` will repeatedly push to `bribesInPool[poolIdentifier]` .

Medium

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PenpieBribeManager.sol#L167-L201

```
function addBribeNative(uint256 _pid) external payable nonReentrant whenNotPaused onlyInEpoch {
    if (msg.value == 0) revert InvalidBribeToken();

    if (_pid >= pools.length) revert InvalidPool();
    Pool memory bribePool = pools[_pid];

    if (!bribePool._active) revert InvalidPool();

    uint256 fee = msg.value * feeRatio / DENOMINATOR;
    uint256 afterFee = msg.value - fee;

    bool success;
    if (fee > 0) {
        (success, ) = distributor.call{value: afterFee}("");
        if (feeCollector == address(0)) {
            unCollectedFee[NATIVE] += fee;
        } else {
            feeCollector.transfer(fee);
        }
    } else {
        (success, )  = distributor.call{value: afterFee}("");
    }

    if (!success) revert InvalidBribeToken();

    // We will generate a unique index for each pool and reward based on the epoch
    bytes32 poolIdentifier = _getPoolIdentifier(currentEpoch, _pid);
    bytes32 rewardIdentifier = _getTokenIdentifier(currentEpoch, _pid, NATIVE);

```

```
196     Bribe storage bribe = bribes[rewardIdentifier];
197     bribe._amount += afterFee;
198     if(bribe._token == address(0)) bribesInPool[poolIdentifier].push(rewardIdentifier);
199
200     emit NewBribe(msg.sender, currentEpoch, _pid, NATIVE, afterFee);
201 }
```

At L198, `bribe._token` is not updated at the same time it is pushed to `bribesInPool[poolIdentifier]`. This causes `addBribeNative()` to repeatedly push to `bribesInPool[poolIdentifier]`.

For reference, `addBribeERC20()` is implemented correctly:

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PenpieBribeManager.sol#L203-L237

```
203 function addBribeERC20(uint256 _pid, address _token, uint256 _amount) external nonReentrant whenNotPaused onlyInEpoch {
204     if (_pid >= pools.length) revert InvalidPool();
205     Pool memory bribePool = pools[_pid];
206
207     if (!bribePool._active) revert InvalidPool();
208     if (!allowedToken[_token]) revert InvalidBribeToken();
209
210     uint256 fee = _amount * feeRatio / DENOMINATOR;
211     uint256 afterFee = _amount - fee;
212
213     // transfer the token to the target directly to save the gas fee
214     if (fee > 0) {
215         IERC20(_token).safeTransferFrom(msg.sender, distributor, afterFee);
216         if (feeCollector == address(0)) {
217             unCollectedFee[_token] += fee;
218             IERC20(_token).safeTransferFrom(msg.sender, address(this), fee);
219         } else {
220             IERC20(_token).safeTransferFrom(msg.sender, feeCollector, fee);
221         }
222     } else {
223         IERC20(_token).safeTransferFrom(msg.sender, distributor, afterFee);
```

```
    }

    bytes32 poolIdentifier = _getPoolIdentifier(currentEpoch, _pid);
    bytes32 rewardIdentifier = _getTokenIdentifier(currentEpoch, _pid, _token);

    Bribe storage bribe = bribes[rewardIdentifier];
    bribe._amount += afterFee;
    if(bribe._token == address(0)) {
        bribe._token = _token;
        bribesInPool[poolIdentifier].push(rewardIdentifier);
    }

    emit NewBribe(msg.sender, currentEpoch, _pid, _token, afterFee);
}
```

Status

✓ **Fixed**

[WP-N7] `VLPenPie` Unconventional ERC20 implemenation, lack of Transfer events in mint (lock) and burn (unlock)

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/171766bda4b95257bf00bb97f2887110f5b1845c/contracts/VLPenPie.sol#L436-L450

```
    function _unlock(uint256 _unlockedAmount) internal {
        IMasterPenpie(masterPenpie).withdrawVlPenpieFor(_unlockedAmount, msg.sender); // trigers update pool share, so happens before total amount reducing
        totalAmountInCoolDown -= _unlockedAmount;
        totalAmount -= _unlockedAmount;
    }

    function _lock(
        address spender,
        address _for,
        uint256 _amount
    ) internal {
        penpie.safeTransferFrom(spender, address(this), _amount);
        IMasterPenpie(masterPenpie).depositVlPenpieFor(_amount, _for);
        totalAmount += _amount; // trigers update pool share, so happens after toal amount increase
    }
```

Status

ⓘ Acknowledged

[WP-N8] Misleading modifier name: `onlyWhenEnd`

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PenpieBribeManager.sol#L105-L109

```
105 modifier onlyWhenEnd() {
106     uint256 epochEndTime = epochStartTime + epochPeriod;
107     if (block.timestamp >= epochStartTime && block.timestamp <= epochEndTime)
    revert OnlyWhenEnd();
108     _;
109 }
```

Should be renamed to `onlyNotInEpoch` .

Status

✓ Fixed

[WP-L9] Unused code

Low

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerBaseUpg.sol#L239-L245

```
function _forwardRewards(address rewardToken, uint256[] memory feeAmounts) internal {
    for (uint256 i; i < feeAmounts.length; i++) {
        if (rewardToken != address(0) && feeAmounts[i] > 0) {
            IERC20(rewardToken).safeTransfer(msg.sender, feeAmounts[i]);
        }
    }
}
```

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerBaseUpg.sol#L92-L97

```
modifier refundUnusedEth() {
    _;
    if (address(this).balance > 0) {
        AddressUpgradeable.sendValue(payable(msg.sender), address(this).balance);
    }
}
```

Status

✓ Fixed

[WP-G10] `removeAllowedTokens()` can be optimized

Gas

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PenpieBribeManager.sol#L297-L313

```
function removeAllowedTokens(address _token) external onlyOwner {
    if (!allowedToken[_token]) revert InvalidBribeToken();

    uint256 i = 0;
    while (allowedTokens[i] != _token) {
        i++;
        if (i >= allowedTokens.length) revert InvalidBribeToken();
    }

    while (i < allowedTokens.length - 1) {
        allowedTokens[i] = allowedTokens[i + 1];
        i++;
    }
    allowedTokens.pop();

    allowedToken[_token] = false;
}
```

Recommendation

Given that the order of `allowedTokens` is not used, consider changing to:

```
function removeAllowedTokens(address _token) external onlyOwner {
    if (!allowedToken[_token]) revert InvalidBribeToken();
    uint256 allowedTokensLength = allowedTokens.length;
    uint256 i = 0;
    while (allowedTokens[i] != _token) {
        i++;
        if (i >= allowedTokensLength) revert InvalidBribeToken();
    }
```

```

    allowedTokens[i] = allowedTokens[allowedTokensLength-1];
    allowedTokens.pop();

    allowedToken[_token] = false;
}
```

Status

✓ Fixed

[WP-L11] When `bribeManager` is updated after initialization, unclaimed past native rewards cannot be claimed.

Low

Issue Description

If someone tries to claim rewards with `_token = oldBribeManager` at line 58, the condition will be true and it will be treated as an ERC20, resulting in a failed claim.

The previously used `claimed[_token][_account]` account will no longer be applicable.

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PenpieBribeRewardDistributor.sol#L126-L171

```
126 function _claim(
127     address _token,
128     address _account,
129     uint256 _amount,
130     bytes32[] calldata _merkleProof
131 ) private {
@@ 132,148 @@
149
150     // Calculate the claimable amount based off the total of reward (used in the merkle tree)
151     // since the beginning for the user, minus the total claimed so far
152     uint256 claimable = _amount - claimed[_token][_account];
153     // Update the claimed amount to the current total
154     claimed[_token][_account] = _amount;
155
156     // Check whether the reward is in the form of native tokens or ERC20
157     // by checking if the token address is set to the bribe vault or not
158     if (_token != bribeManager) {
159         IERC20(_token).safeTransfer(_account, claimable);
160     } else {
161         (bool sent, ) = payable(_account).call{value: claimable}("");
162         if(!sent) revert TransferFailed();
163     }
164
```

```
    emit RewardClaimed(
        _token,
        _account,
        claimable,
        reward.updateCount
    );
}
```

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PenpieBribeRewardDistributor.sol#L205-L207

```
function setBribeManager(address _manager) external onlyOwner {
    bribeManager = _manager;
}
```

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PenpieBribeRewardDistributor.sol#L92-L95

```
modifier onlyBribeManager() {
    if (msg.sender != bribeManager) revert OnlyBribeManager();
    _;
}
```

Recommendation

Instead of using `bribeManager` as the token address for the native token, consider using the same native token address (`address(1)`) as other contracts in the system:

```
if (_token != NATIVE) {
    IERC20(_token).safeTransfer(_account, claimable);
} else {
    (bool sent, ) = payable(_account).call{value: claimable}("");
    if(!sent) revert TransferFailed();
}
```

`emergencyWithdraw()` should also be updated accordingly.

`bribeManager` can be removed from the `PenpieBribeRewardDistributor` contract.

Status

✓ **Fixed**

[WP-L12] Wrong implemenation of `vePendlePerLockedPenpie()`

Low

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L65-L69

```
function vePendlePerLockedPenpie() public view returns (uint256) {
    if (IVLPenpie(vlPenpie).totalLocked() == 0) return 0;
    uint256 multiplier = _getMultiplier();
    return (totalVotes() * 1e18 * multiplier) / IVLPenpie(vlPenpie).totalLocked();
}
```

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L169-L176

```
function _getMultiplier() internal view returns(uint256){
    uint256 multiplier = 1;
    uint256 exactValue = IVLPenpie(vlPenpie).totalLocked() / 1e18;
    while (exactValue / multiplier >= 10) {
        multiplier *= 10;
    }
    return multiplier;
}
```

Based on the function name, it seems like there is no need to include a `multiplier` in the formula.

Recommendation

```
function vePendlePerLockedPenpie() public view returns (uint256) {
    if (IVLPenpie(vlPenpie).totalLocked() == 0) return 0;
```

```
    return totalVotes() * 1e18 / IVLPenpie(vlPenpie).totalLocked();
}
```

Status

✓ Fixed

[WP-H13] `PendleVoteManagerMainChain#castVotes()` will revert when `PendleVoteManagerSideChain` casts their collective votes via LayerZero.

High

Issue Description

As there are no special treatments for `PendleVoteManagerSideChain` in `getUserVotable()`, `_updateVoteAndCheck()` on line 213 will revert with an error `NotEnoughVote`, effectively preventing vlPenpie holders on the side chain from exercising their voting rights.

https://github.com/magpiexyz/pendleMagpie/blob/f4a88eef3e14d0b93e2d9e298585d59c29be305a/contracts/bribeMarket/PendleVoteManagerSideChain.sol#L80-L108

```
80  function castVotes() override public payable
81  {
82      lastCastTime = block.timestamp;
83      uint256 length = poolInfos.length;
84      UserVote[] memory votes = new UserVote[](length);
85
86      for (uint16 i; i < length; i++) {
87          votes[i].pid = i;
88          votes[i].weight = deltaSinceLastCast[i];
89
90          deltaSinceLastCast[i] = 0; // might need a safer way to deal with this
91      }
92
93      uint minDstGas = minDstGasLookup[LayerZeroHelper._getLayerZeroChainId(mainChainId)][1];
94      if (minDstGas == 0 || minRemoteCastGas < minDstGas) revert RemoteMinGasNotSet();
95
96      bytes memory lzAdapater = abi.encodePacked(uint16(1), minRemoteCastGas);
97
98      _lzSend (
99          LayerZeroHelper._getLayerZeroChainId(mainChainId),
100         encodeVote(address(this), votes),
```

```
        payable(msg.sender),
        address(0),
        lzAdapater,
        msg.value
    );

    emit VoteCasted(msg.sender, lastCastTime);
}
```

https://github.com/magpiexyz/pendleMagpie/blob/f4a88eef3e14d0b93e2d9e298585d59c29be305a/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L135-L143

```
function _nonblockingLzReceive(uint16 _srcChainId, bytes memory _srcAddress, uint64 _nonce, bytes memory _payload) internal override {
    (address user, UserVote[] memory userVotes) = decodeVote(_payload);
    if (remotePendleVoter[user])
        _recCast(user, userVotes);
}

function _recCast(address _user, UserVote[] memory _userVotes) internal {
    _updateVoteAndCheck(_user, _userVotes);
}
```

https://github.com/magpiexyz/pendleMagpie/blob/f4a88eef3e14d0b93e2d9e298585d59c29be305a/contracts/bribeMarket/PendleVoteManagerBaseUpg.sol#L177-L214

```
function _updateVoteAndCheck(address _user, UserVote[] memory _userVotes) internal {
    uint256 length = _userVotes.length;
    int256 totalUserVote;

    for (uint256 i; i < length; i++) {
        Pool storage pool = poolInfos[_userVotes[i].pid];

        int256 weight = _userVotes[i].weight;
        totalUserVote += weight;

```

```
        if (weight != 0) {
            if (weight > 0) {
                // Prevent users increase voting in discarded pools
                if (!pool.isActive) revert PoolNotActive();
                uint256 absVal = uint256(weight);
                pool.totalVoteInVlPenpie += absVal;
                userVotedForPoolInVlPenpie[_user][pool.market] += absVal;
            } else {
                uint256 absVal = uint256(-weight);
                pool.totalVoteInVlPenpie -= absVal;
                userVotedForPoolInVlPenpie[_user][pool.market] -= absVal;
            }
        }

        _afterVoteUpdate(_user, pool.market, _userVotes[i].pid, weight);
    }

    // update user's total vote and all vlPNP vote
    if (totalUserVote > 0) {
        userTotalVotedInVlPenpie[_user] += uint256(totalUserVote);
        totalVlPenpieInVote += uint256(totalUserVote);
    } else {
        userTotalVotedInVlPenpie[_user] -= uint256(-totalUserVote);
        totalVlPenpieInVote -= uint256(-totalUserVote);
    }

    if (userTotalVotedInVlPenpie[_user] > getUserVotable(_user)) revert NotEnoughVote();
}
```

https://github.com/magpiexyz/pendleMagpie/blob/f4a88eef3e14d0b93e2d9e298585d59c29be305a/contracts/bribeMarket/PendleVoteManagerBaseUpg.sol#L103-L105

```
function getUserVotable(address _user) public view returns (uint256) {
    return IVLPenpie(vlPenpie).getUserTotalLocked(_user);
}
```

Recommendation

Consider skipping the `PendleVoteManagerBaseUpg._updateVoteAndCheck()` check on line 213 for addresses with `remotePendleVoter[user]` , which verifies that `userTotalVotedInVlPenpie[_user] <= getUserVotable(_user)` .

Status

✓ Fixed

[WP-H14] `PendleVoteManagerSideChain.castVotes()` can be blocked due to requirement in `PendleVoteManagerBaseUpg._updateVoteAndCheck()`

High

Issue Description

The requirement of `pool.isActive || weight <= 0` in `PendleVoteManagerBaseUpg._updateVoteAndCheck()` at L190 may cause `PendleVoteManagerSideChain.castVotes()` to be blocked.

This is because for a pool with `!pool.isActive`, its `PendleVoteManagerSideChain.deltaSinceLastCast[pid]` may be greater than 0.

However, `PendleVoteManagerMainChain._nonblockingLzReceive()` requires that the weight of a pool with `!pool.isActive` must be <= 0 when receiving `votes`, otherwise it will `revert PoolNotActive()`.

https://github.com/magpiexyz/pendleMagpie/blob/f4a88eef3e14d0b93e2d9e298585d59c29be305a/contracts/bribeMarket/PendleVoteManagerSideChain.sol#L78-L108

```
78      /// @notice cast all pending votes back to Eth
79      /// @notice this function will be gas intensive, hence a fee is given to the caller
80      function castVotes() override public payable
81      {
82          lastCastTime = block.timestamp;
83          uint256 length = poolInfos.length;
84          UserVote[] memory votes = new UserVote[](length);
85
86          for (uint16 i; i < length; i++) {
87              votes[i].pid = i;
88              votes[i].weight = deltaSinceLastCast[i];
89
90              deltaSinceLastCast[i] = 0; // might need a safer way to deal with this
91          }
92
```

```
        uint minDstGas = minDstGasLookup[LayerZeroHelper._getLayerZeroChainId(mainChainId)][1];
        if (minDstGas == 0 || minRemoteCastGas < minDstGas) revert RemoteMinGasNotSet();

        bytes memory lzAdapater = abi.encodePacked(uint16(1), minRemoteCastGas);

        _lzSend (
            LayerZeroHelper._getLayerZeroChainId(mainChainId),
            encodeVote(address(this), votes),
            payable(msg.sender),
            address(0),
            lzAdapater,
            msg.value
        );

        emit VoteCasted(msg.sender, lastCastTime);
    }
```

https://github.com/magpiexyz/pendleMagpie/blob/f4a88eef3e14d0b93e2d9e298585d59c29be305a/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L135-L143

```
    function _nonblockingLzReceive(uint16 _srcChainId, bytes memory _srcAddress, uint64 _nonce, bytes memory _payload) internal override {
        (address user, UserVote[] memory userVotes) = decodeVote(_payload);
        if (remotePendleVoter[user])
            _recCast(user, userVotes);
    }

    function _recCast(address _user, UserVote[] memory _userVotes) internal {
        _updateVoteAndCheck(_user, _userVotes);
    }
```

https://github.com/magpiexyz/pendleMagpie/blob/f4a88eef3e14d0b93e2d9e298585d59c29be305a/contracts/bribeMarket/PendleVoteManagerBaseUpg.sol#L177-L214

```
    function _updateVoteAndCheck(address _user, UserVote[] memory _userVotes) internal {
        uint256 length = _userVotes.length;
        int256 totalUserVote;

        for (uint256 i; i < length; i++) {
            Pool storage pool = poolInfos[_userVotes[i].pid];

            int256 weight = _userVotes[i].weight;
            totalUserVote += weight;

            if (weight != 0) {
                if (weight > 0) {
                    // Prevent users increase voting in discarded pools
                    if (!pool.isActive) revert PoolNotActive();
                    uint256 absVal = uint256(weight);
                    pool.totalVoteInVlPenpie += absVal;
                    userVotedForPoolInVlPenpie[_user][pool.market] += absVal;
                } else {
                    uint256 absVal = uint256(-weight);
                    pool.totalVoteInVlPenpie -= absVal;
                    userVotedForPoolInVlPenpie[_user][pool.market] -= absVal;
                }
            }

            _afterVoteUpdate(_user, pool.market, _userVotes[i].pid, weight);
        }

        // update user's total vote and all vlPNP vote
        if (totalUserVote > 0) {
            userTotalVotedInVlPenpie[_user] += uint256(totalUserVote);
            totalVlPenpieInVote += uint256(totalUserVote);
        } else {
            userTotalVotedInVlPenpie[_user] -= uint256(-totalUserVote);
            totalVlPenpieInVote -= uint256(-totalUserVote);
        }

        if (userTotalVotedInVlPenpie[_user] > getUserVotable(_user)) revert NotEnoughVote();
    }
```

Recommendation

- Change the denominator of `PendleVoteManagerMainChain.castVotes()` L124 to the newly calculated `totalActiveVoteInVlPenpie` instead of the storage value, to prevent inactive pools with `!pool.isActive` from impacting the distribution of Pendle votes in `IPendleStaking(pendleStaking).vote(_pools, votes)`
- Remove the `pool.isActive || weight <= 0` condition in `PendleVoteManagerBaseUpg._updateVoteAndCheck()` L190.

Status

✓ **Fixed**

[WP-L15] `PendleVoteManagerMainChain.castVotes()` should not be `payable`

Low

Issue Description

`PendleVoteManagerMainChain.castVotes()` has a `payable` modifier, but there is no usage of `msg.value`, hence it should not be `payable`.

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L105-L136

```
    /// @notice cast all pending votes
    /// @notice this function will be gas intensive, hence a fee is given to the caller
    function castVotes() override public payable
    {
        lastCastTime = block.timestamp;
        uint256 length = poolInfos.length;

        address[] memory _pools = new address[](length);
        uint64[] memory votes = new uint64[](length);

        for (uint256 i; i < length; i++) {
            Pool storage pool = poolInfos[i];
            _pools[i] = pool.market;

            uint256 currentVote = getVoteForMarket(pool.market);
            uint256 targetVoteInVlPenpie = pool.totalVoteInVlPenpie;
            uint256 targetVote = 0;

            if (totalVlPenpieInVote != 0) {
                targetVote =(targetVoteInVlPenpie * totalVotes()) / totalVlPenpieInVote;
            }

            if (targetVote >= currentVote)
                votes[i] = _getVoteInPercentage(int256(targetVote - currentVote), currentVote);
```

```
            else
                votes[i] = _getVoteInPercentage(int256(targetVote) - int256(currentVote), currentVote);

        }

        IPendleStaking(pendleStaking).vote(_pools, votes);
        emit VoteCasted(msg.sender, lastCastTime);
    }
```

Recommendation

Consider removing the `payable` modifier.

Status

✓ Fixed

[WP-N16] Mismatch between comment and implementation.

Issue Description

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerMainChain.sol#L105-L136

```
105 /// @notice cast all pending votes
106 /// @notice this function will be gas intensive, hence a fee is given to the caller
107 function castVotes() override public payable
108 {
109     lastCastTime = block.timestamp;
110     uint256 length = poolInfos.length;
111
112     address[] memory _pools = new address[](length);
113     uint64[] memory votes = new uint64[](length);
114
115     for (uint256 i; i < length; i++) {
116         Pool storage pool = poolInfos[i];
117         _pools[i] = pool.market;
118
119         uint256 currentVote = getVoteForMarket(pool.market);
120         uint256 targetVoteInVlPenpie = pool.totalVoteInVlPenpie;
121         uint256 targetVote = 0;
122
123         if (totalVlPenpieInVote != 0) {
124             targetVote =(targetVoteInVlPenpie * totalVotes()) / totalVlPenpieInVote;
125         }
126
127         if (targetVote >= currentVote)
128             votes[i] = _getVoteInPercentage(int256(targetVote - currentVote), currentVote);
129         else
130             votes[i] = _getVoteInPercentage(int256(targetVote) - int256(currentVote), currentVote);
131
132     }
133
134     IPendleStaking(pendleStaking).vote(_pools, votes);
```

```
    emit VoteCasted(msg.sender, lastCastTime);
}
```

While the comment states that:

> a fee is given to the caller

There is no such feature in the implementation.

Status

✓ **Fixed**

[WP-I17] Storage Gaps at concrete contract storage layout bottom is useless

Informational

Issue Description

Consider moving the `__gap` to the top of the concrete contract storage layout as an empty reserved space in storage that can be used to add addtional parent in future implemenation version.

See: https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps

https://github.com/magpiexyz/pendleMagpie/blob/a5c037f36b4d3118f5722f075fb398efc989f0ee/contracts/bribeMarket/PendleVoteManagerSideChain.sol#L15-L135

```
@@ 15,21 @@
contract PendleVoteManagerSideChain is PendleVoteManagerBaseUpg {
    using SafeERC20 for IERC20;

    /* ============ Structs ============ */

    /* ============ State Variables ============ */

    mapping(uint256 => int256) public deltaSinceLastCast; // pid -> delta
    uint256 public mainChainId; // EVM chain Id, NOT layerZero chainId
    uint256 public minRemoteCastGas;

    uint256[50] private __gap;

@@ 35,134 @@
}
```

Recommendation

Consider chainging to:

```
@@ 15,21 @@
contract PendleVoteManagerSideChain is PendleVoteManagerBaseUpg {
    using SafeERC20 for IERC20;

    /* ============ Structs ============ */

    /* ============ State Variables ============ */

    uint256[1000] private __gap;

    mapping(uint256 => int256) public deltaSinceLastCast; // pid -> delta
    uint256 public mainChainId; // EVM chain Id, NOT layerZero chainId
    uint256 public minRemoteCastGas;

@@ 35,134 @@
}
```

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.