SHERLOCK

SHERLOCK SECURITY REVIEW FOR

PERENNIAL

Prepared For: Perennial

Prepared By: Sherlock

Lead Security Expert: WatchPug

Dates: February 9th - 14th, 2023

Introduction

“Perennial is a cash-settled synthetic derivatives protocol. It allows developers to launch any synthetic market with just a few lines of code.”

This report is a follow-up security review for Perennial Protocol that was prepared by WatchPug from February 9th - 14th, 2023.

Scope

Scope: Main review focus, small update #1 (Vault fixes), small update #2 (Add compound VAULT_WRAP_AND_DEPOSIT action)

Protocol Info

Language: Solidity

Blockchain: Ethereum
L2s: None

Tokens used: USDC, DSU, Reward ERC20 tokens

Findings

Each issue has an assigned severity:

- Informational issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgement as to whether to address such issues.
- Low issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Total Issues

Informational	Low	Medium	High
1	1	1	0

Issue M-01

_rebalancePosition() will rebalance to a incorrect target position due to incorrect calculation of currentAssets

Summary

_rebalancePosition() will rebalance to a incorrect target position due to incorrect calculation of currentAssets. See code snippets and POC below for more detail.

Severity

Medium

Issue Detail

https://github.com/equilibria-xyz/perennial-mono/blob/ea722b43df29b9693f4b4e8ebecb31d8258dbe1b/packages/perennial-vaults/contracts/BalancedVault.sol#L400-L412

```
400 function _rebalancePosition(VersionContext memory context, UFixed18 claimAmount) private {
401     UFixed18 currentAssets = _totalAssetsAtVersion(context).add(_deposit).sub(claimAmount);
402     if (currentAssets.lt(controller.minCollateral().mul(TWO))) currentAssets = UFixed18Lib.ZERO;
403
404     UFixed18 currentUtilized = _totalSupply.add(_redemption).isZero() ?
405         currentAssets :
406         currentAssets.muldiv(_totalSupply, _totalSupply.add(_redemption));
407     UFixed18 currentPrice = long.atVersion(context.version).price.abs();
408     UFixed18 targetPosition = currentUtilized.mul(targetLeverage).div(currentPrice).div(TWO);
409
410     _updateMakerPosition(long, targetPosition);
411     _updateMakerPosition(short, targetPosition);
412 }
```

https://github.com/equilibria-xyz/perennial-mono/blob/ea722b43df29b9693f4b4e8ebecb31d8258dbe1b/packages/perennial-vaults/contracts/BalancedVault.sol#L578-L583

```
578 function _totalAssetsAtVersion(VersionContext memory context) private view returns (UFixed18) {
579     (UFixed18 longCollateral, UFixed18 shortCollateral, UFixed18 idleCollateral) = _collateral();
580     (UFixed18 totalCollateral, UFixed18 totalDebt) =
581         (longCollateral.add(shortCollateral).add(idleCollateral), _totalUnclaimedAtVersion(context).add(_deposit));
582     return totalCollateral.gt(totalDebt) ? totalCollateral.sub(totalDebt) : UFixed18Lib.ZERO;
583 }
```

Tool used

Manual Review

Proof of Concept

Given:

- longCollateral == 150 (50 of which belongs to exitsing stakeholders and 100 to the new _deposit)
- shortCollateral == 150 (same as above)
- idleCollateral == 0
- _deposit == 200
- totalShares == 100

Alice redeem() 40 shares;

- totalShares: 100 → 60
- _redemption: 0 → 40

L170, _rebalance():

- _totalAssetsAtVersion() ⇒ (150 + 150 + 0) - (0 + 200) == 100
- L401, currentAssets ⇒ 100 + 200 == 300;
- L406, currentUtilized ⇒ 300 * 60 / 100 == 180;
- L408, targetPosition ⇒ 180 * targetLeverage / currentPrice / 2

The expected result is that all of the _deposit (200) should be invested, with 60 of the existing position collateral (100) kept and only 40 removed.

So the targetPosition should be 260 * targetLeverage / currentPrice / 2.

Recommendation

L401 currentAssets should not include _deposit; _deposit should be added to currentUtilized at L408 instead:

```
function _rebalancePosition(VersionContext memory context, UFixed18 claimAmount) private {
    UFixed18 currentAssets = _totalAssetsAtVersion(context).sub(claimAmount);
    if (currentAssets.lt(controller.minCollateral().mul(TWO))) currentAssets = UFixed18Lib.ZERO;

    UFixed18 currentUtilized = _totalSupply.add(_redemption).isZero() ?
        currentAssets :
        currentAssets.muldiv(_totalSupply, _totalSupply.add(_redemption));
    UFixed18 currentPrice = long.atVersion(context.version).price.abs();
    UFixed18 targetPosition = (currentUtilized.add(_deposit)).mul(targetLeverage).div(currentPrice).div(TWO);

    _updateMakerPosition(long, targetPosition);
    _updateMakerPosition(short, targetPosition);
}
```

Perennial Comment

Fixed via https://github.com/equilibria-xyz/perennial-mono/pull/134

WatchPug Comment

The fix of M-01 is good.

Issue L-01

The portion of funds corresponding to the exited shares (_totalUnclaimed) should be moved from products' collateral to balance in order to avoid potential damage

Severity
Low

Issue Detail
https://github.com/equilibria-xyz/perennial-mono/blob/ea722b43df29b9693f4b4e8ebecb31d8258dbe1b/packages/perennial-vaults/contracts/BalancedVault.sol#L384-L395

```
384 function _rebalanceCollateral(UFixed18 claimAmount) private {
385     (UFixed18 longCollateral, UFixed18 shortCollateral, UFixed18 idleCollateral) = _collateral();
386     UFixed18 currentCollateral = longCollateral.add(shortCollateral).add(idleCollateral).sub(claimAmount);
387     UFixed18 targetCollateral = currentCollateral.div(TWO);
388     if (targetCollateral.lt(controller.minCollateral())) targetCollateral = UFixed18Lib.ZERO;
389
390     (IProduct greaterProduct, IProduct lesserProduct) =
391         longCollateral.gt(shortCollateral) ? (long, short) : (short, long);
392
393     _updateCollateral(greaterProduct, greaterProduct == long ? longCollateral : shortCollateral, targetCollateral);
394     _updateCollateral(lesserProduct, lesserProduct == long ? longCollateral : shortCollateral, targetCollateral);
395 }
```

_totalUnclaimed should be excluded from currentCollateral, so that it won't be deposited to the products as collateral.

In the current implementation, exited shares become free liquidity recorded as _totalUnclaimed, but the _totalUnclaimed isn't removed from the collateral to the balance of the BalancedVault. As a result, future pnl can potentially damage _totalUnclaimed.

In the most extreme case, if one exits but never comes to claim it in the next version, a future liquidation can potentially make the user unable to get back their unclaimed funds.

Recommendation
Change to:

https://github.com/equilibria-xyz/perennial-mono/blob/ea722b43df29b9693f4b4e8ebecb31d8258dbe1b/packages/perennial-vaults/contracts/BalancedVault.sol#L384-L395

```
384 function _rebalanceCollateral(UFixed18 claimAmount) private {
385     (UFixed18 longCollateral, UFixed18 shortCollateral, UFixed18 idleCollateral) = _collateral();
386     UFixed18 currentCollateral = longCollateral.add(shortCollateral).add(idleCollateral).sub(claimAmount);
387     UFixed18 targetCollateral;
388     if (currentCollateral.lt(_totalUnclaimed)) {
389         targetCollateral = UFixed18Lib.ZERO;
390     } else {
391         targetCollateral = currentCollateral.sub(_totalUnclaimed).div(TWO);
392         if (targetCollateral.lt(controller.minCollateral())) targetCollateral = UFixed18Lib.ZERO;
393     }
394
395     (IProduct greaterProduct, IProduct lesserProduct) =
396         longCollateral.gt(shortCollateral) ? (long, short) : (short, long);
397
398     _updateCollateral(greaterProduct, greaterProduct == long ? longCollateral : shortCollateral, targetCollateral);
399     _updateCollateral(lesserProduct, lesserProduct == long ? longCollateral : shortCollateral, targetCollateral);
400 }
```

Perennial Comment

Acknowledged, but fixing will cause secondary issues.

In the cases of extreme price changes like the above, the fix may cause a situation where targetCollateral becomes zero, but we still have positions open. In this case, attempting to retarget the collateral will revert because the position retargeting won't be settled until the following version. By leaving the collateral in the markets, we gain the ability to at least sync the vault to recapitalize and retarget everything properly.

WatchPug Comment

Agreed on not fixing.

Issue I-01

New deposit is added as collateral and start bearing the potential damage from existing positions while its corresponding positions have not yet been created

Severity
Informational

Issue Detail
In order to expand the positions, new deposit is added as collateral immediately.

As a side effect, while the additional positions haven't been created yet, if the positions go insolvent, the new deposit will be damaged.

https://github.com/equilibria-xyz/perennial-mono/blob/ea722b43df29b9693f4b4e8ebecb31d8258dbe1b/packages/perennial-vaults/contracts/BalancedVault.sol#L384-L396

```
384 function _rebalanceCollateral(UFixed18 claimAmount) private {
385     (UFixed18 longCollateral, UFixed18 shortCollateral, UFixed18 idleCollateral) = _collateral();
386     UFixed18 currentCollateral = longCollateral.add(shortCollateral).add(idleCollateral).sub(claimAmount);
387     UFixed18 targetCollateral = currentCollateral.div(TWO);
388     if (targetCollateral.lt(controller.minCollateral())) targetCollateral = UFixed18Lib.ZERO;
389
390     (IProduct greaterProduct, IProduct lesserProduct) =
391         longCollateral.gt(shortCollateral) ? (long, short) : (short, long);
392
393     _updateCollateral(greaterProduct, greaterProduct == long ? longCollateral : shortCollateral, targetCollateral);
394     _updateCollateral(lesserProduct, lesserProduct == long ? longCollateral : shortCollateral, targetCollateral);
395 }
```

Proof of Concept
Given:

- currentPrice: $1000
- currentPosition: (effective size)
    - long: 10 ETH
    - short: 0 (no takers)
- collateral:
    - long: $1000
    - short: $1000

When the user deposits $2000, it will be added as collateral:

- new collateral:
    - long: $2000
    - short: $2000

And it places openMake() orders to expand position sizes.

The price is updated to $800 in the next version.

Before the openMake() orders are executed (position a→b), the PNL from existing position (value a→b) can cause so much damage to the collateral (-$2000) that the whole account could be zeroed out.

To better demonstrate the issue, let's compare it with adding the collateral and creating a set of new positions in a new vault.

There will be no damage done to the collateral, as the new position was just created at the new price of the new version.

Perennial Comment
Acknowledged, but won't fix.

This issue is similar to the above, but on the deposit side. Fortunately on the deposit side, we are guaranteed to only be exposed to this slashing risk for a single oracle version. This means the probability of incurring this slashing is about the same as the underlying product itself going insolvent, which is an acceptable risk profile.

WatchPug Comment
Agreed on not fixing.