Report

v. 1.1

Customer

Lumis

Smart Contract Audit

Lumis Contracts

10th September 2025

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	04.09.25	A. Zveryanskaya	Initial Draft
0.2	04.09.25	A. Zveryanskaya	Minor revision
1.0	04.09.25	A. Zveryanskaya	Release
1.01	10.09.25	D. Khovratovich	Typo fixes

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

src/		
ALM.sol		
src/core/Base/		
Base.sol	BaseStrategyHook.sol	
src/core/		
Oracle.sol	SRebalanceAdapter.sol	
src/core/lendingAdapters/		
EulerLendingAdapter.sol	MorphoLendingAdapter.sol	
src/core/positionManagers/		
PositionManager.sol	UnicordPositionManager.sol	
src/core/swapAdapters/		
UniswapV3SwapAdapter.sol		
src/libraries/		
ALMMathLib.sol	CurrencySettlerSafe.sol	TokenWrapperLib.sol

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Recommendations** contain code style, best practices and other suggestions.

5 Our findings

We found 1 critical, 4 major, and a few less important issues. All identified Critical and Major issues have been fixed.

Fixed 5 out of 5 issues

6 Critical Issues

CVF-1 FIXED

- **Category** Flaw
- **Source** BaseStrategyHook.sol

Description Using the “absSub” function here discards information about which token has more decimals, while this information is crucial for proper price handling.

Recommendation Pass a signed decimals difference into the “getPoolPriceFromOraclePrice” and there do WAD * (10**decimalsDelta) for positive delta and WAD / (10**(-decimalsDelta)) for negative delta.

Client Comment *Now it supports, and we move all price/sqrt price calculations to the oracle.*

```
143 ALMMathLib.getPoolPriceFromOraclePrice(oracle.price(),
        ↪ isInvertedPool, uint8(ALMMathLib.absSub(bDec, qDec)))
```

7 Major Issues

CVF-2 FIXED

- **Category** Unclear behavior
- **Source** ALMMathLib.sol

Description This doesn't support negative decimals delta.

Recommendation Implement proper support for negative decimals delta.

Client Comment *Now it supports, and we move all price/sqrt price calculations to the oracle.*

```
152 uint256 ratio = WAD * (10 ** decimalsDelta); // @Notice: 1e12/p, 1
    ↪ e30 is 1e12 with 18 decimals
```

```
162 uint256 ratio = WAD * (10 ** decimalsDelta); // @Notice: 1e12/p, 1
    ↪ e30 is 1e12 with 18 decimals
```

CVF-3 FIXED

- **Category** Unclear behavior
- **Source** BaseStrategyHook.sol

Description The "onlyAuthorizedPool" modifier doesn't make much sense without the "onlyPoolManager" modifier, as one could call the hook from an invalid address providing a valid pool key as an argument.

Recommendation Add the "onlyPoolManaager" modifier, or just override the "_beforeAddLiquidity" function from the "BaseHook" contract.

Client Comment *The Uniswap team has updated their BaseHook contract, which we inherit, and now all hook functions are protected by onlyPoolManager modifier by default.*

```
125 ) external view override onlyAuthorizedPool(key) returns (bytes4) {
```

CVF-4 FIXED

- **Category** Unclear behavior
- **Source** SRebalanceAdapter.sol

Description In case both amounts were insufficient, the second swap will make “base” amount insufficient again.

Recommendation Ensure quote amount is sufficient after performing the first swap.

Client Comment *Rewritten so only one swap can be used. Didn’t add a check for the other balance because the FL will check it in the next line and revert if not enough.*

197
```
if (amountB > baseBalanceUnwr()) swapAdapter.swapExactOutput(quote,
   ↪ base, amountB - baseBalanceUnwr());
if (amountQ > quoteBalanceUnwr()) swapAdapter.swapExactOutput(base,
   ↪ quote, amountQ - quoteBalanceUnwr());
```

CVF-5 FIXED

- **Category** Flaw
- **Source** ALM.sol

Description This code is vulnerable for reentrancy attack.

Recommendation Set stored fee amounts to zero before calling external contracts.

Client Comment *We were forced to split the contract into ALM and BaseStrategyHook due to size limits. Now, all assets in the whole contract can only be fees, so we withdraw the full balance without stored fee amounts.*

293
```
IERC20(base).safeTransfer(treasury, accumulatedFeeB);
IERC20(quote).safeTransfer(treasury, accumulatedFeeQ);
accumulatedFeeB = 0;
accumulatedFeeQ = 0;
```

8 Moderate Issues

CVF-6 FIXED

- **Category** Procedural
- **Source** ALMMathLib.sol

Description Importing “test/utils” into production code seems weird.

Recommendation Avoid using test utilities in production.

Client Comment *Now we use the v4-periphery/src/libraries/LiquidityAmounts.sol*

```
7 import {LiquidityAmounts} from "v4-core/../test/utils/
    ↪ LiquidityAmounts.sol";
```

CVF-7 FIXED

- **Category** Overflow/Underflow
- **Source** ALMMathLib.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculation overflows.

Recommendation User the “mulDiv” function.

Client Comment *Some formulas are gone, the rest is fixed.*

```
22 uint160 sqrtPriceDeltaX96 = SafeCast.toUint160((amount1 * Q96) /
    ↪ liquidity);
```

```
31 uint160 sqrtPriceDeltaX96 = SafeCast.toUint160((amount1 * Q96) /
    ↪ liquidity);
```

```
42         uint256(liquidity).mul(uint256(sqrtPriceCurrentX96)).div(
               uint256(liquidity) - amount0.mul(uint256(
                   ↪ sqrtPriceCurrentX96)).div(Q96)
```

```
55         uint256(liquidity).mul(uint256(sqrtPriceCurrentX96)).div(
               uint256(liquidity) + amount0.mul(uint256(
                   ↪ sqrtPriceCurrentX96)).div(Q96)
```

```
79 else return (ts.mul(TVL2 - TVL1)).div(TVL1);
```

```
97          ? SafeCast.toUint256((baseValue * SafeCast.toInt256(price))
               ↪ / int256(WAD) + variableValue)
            : SafeCast.toUint256(baseValue + (variableValue * int256(WAD
               ↪ )) / SafeCast.toInt256(price));
```

```
108     (SafeCast.toInt256(weight) * (SafeCast.toInt256(longLeverage) -
           ↪ SafeCast.toInt256(shortLeverage))) /
            int256(WAD) +
```

```
144 return uint256(sqrtPriceX96).pow(2 * WAD).mul(WAD * WAD).div(Q192);
```

CVF-8 FIXED

- **Category** Flaw
- **Source** ALMMathLib.sol

Description Here the “ratio” value, which is result of a division, is used in multiplication. This is a bad practice as it multiplies rounding errors.

Recommendation Do division after multiplication.

Client Comment *Now use mulDiv 4 times.*

```
127 uint256 ratio = sharesOut.div(totalSupply);
    return (collateralLong.mul(ratio), collateralShort.mul(ratio),
       ↪ debtLong.mul(ratio), debtShort.mul(ratio));
```

CVF-9 FIXED

- **Category** Suboptimal
- **Source** ALMMathLib.sol

Description Using the “pow” function to square a number is very inefficient.

Recommendation Just multiply the number by itself.

Client Comment *The formula is rewritten in a new way, without the ”pow”.*

```
144 return uint256(sqrtPriceX96).pow(2 * WAD).mul(WAD * WAD).div(Q192);
```

CVF-10 FIXED

- **Category** Bad datatype
- **Source** TokenWrapperLib.sol

Description The “WAD” constant should be used instead of “18”.

Client Comment *We have WAD constant for 1e18, will use WAD_DECIMALS instead.*

```
9  else if (tokenWAD > WAD) return a / 10 ** (tokenWAD - 18);
10 else return a * 10 ** (18 - tokenWAD);
```

```
15 else if (tokenWAD > WAD) return a * 10 ** (tokenWAD - 18);
   else return a / 10 ** (18 - tokenWAD);
```

CVF-11 FIXED

- **Category** Suboptimal
- **Source** UniswapV3SwapAdapter.sol

Description Hardcoding mainnet addresses makes code harder to test.

Recommendation Pass the router address as a constructor argument and store in an immutable variable.

```
20 address constant SWAP_ROUTER = 0
      ↪ xE592427A0AEce92De3Edee1F18E0157C05861564;
```

CVF-12 INFO

- **Category** Suboptimal
- **Source** UniswapV3SwapAdapter.sol

Description Using a router, when the particular pool address is known, is waste of gas.

Recommendation Call the target pool directly.

Client Comment *We decided to go in the opposite direction and integrate Uniswap Universal Router 2, which allows us to swap across various V2-V4 pools.*

```
38     ISwapRouter.ExactInputSingleParams({
```

```
41         fee: IUniswapV3Pool(targetPool).fee(),
```

```
59 ISwapRouter.ExactOutputSingleParams({
```

```
62     fee: IUniswapV3Pool(targetPool).fee(),
```

CVF-13 FIXED

- **Category** Suboptimal
- **Source** UniswapV3SwapAdapter.sol

Description Performing two transfers of "tokenIn" is inefficient. Call pool directly, and perform one transfer in a callback.

```
57 IERC20(tokenIn).safeTransferFrom(msg.sender, address(this), IERC20(
   ↪ tokenIn).balanceOf(msg.sender));
```

```
72     IERC20(tokenIn).safeTransfer(msg.sender, IERC20(tokenIn).
          ↪ balanceOf(address(this)));
```

CVF-14 FIXED

- **Category** Suboptimal
- **Source** MorphoLendingAdapter.sol

Description Hardcoding mainnet addresses makes it harder testing the code.

Recommendation Pass the EVC address as a constructor argument and store in an immutable variable.

```
27 IMorpho constant morpho = IMorpho(0
   ↪ xBBBBBbbBBb9cC5e90e3b3Af64bdAF62C37EEFFCb);
```

CVF-15 FIXED

- **Category** Unclear behavior
- **Source** MorphoLendingAdapter.sol

Description If "_earnQuote" is zero, the "_earnBase" argument is silently ignored.

Recommendation Explicitly require the "_earnBase" argument to be zero in case "_earnQuote" is zero.

```
35 if (_earnQuote != address(0)) {
```

CVF-16 FIXED

- **Category** Suboptimal
- **Source** EulerLendingAdapter.sol

Description Hardcoding mainnet addresses makes it harder testing the code.

Recommendation Pass the EVC address as a constructor argument and store in an immutable variable.

```
24 IEVC constant evc = IEVC(0x0C9a3dd6b8F28529d72d7f9cE918D493519EE383)
   ↪ ;
```

CVF-17 FIXED

- **Category** Unclear behavior
- **Source** Base.sol

Description As zero base has a special meaning of “not set”, there should be an explicit check to ensure, that “_base” is not zero.

Recommendation Add an explicit require statement to ensure “_base” is not zero.

Client Comment *Now base/quote set as immutable, so this function is removed.*

```
40 if (base != address(0)) revert TokensAlreadyInitialized();
```

```
42 base = _base;
```

CVF-18 FIXED

- **Category** Unclear behavior
- **Source** Base.sol

Description In case “token” is neither “base” nor “quote” this function silently returns “base”.

Recommendation Revert in such a case.

Client Comment *Function removed.*

```
89 function otherToken(address token) internal view returns (address) {
90     return token == base ? quote : base;
```

CVF-19 FIXED

- **Category** Suboptimal
- **Source** Oracle.sol

Description These values should be precomputed in the constructor and stored in immutable variables.

```
27 uint8 decimalsQuote = feedQuote.decimals();
```

```
29 uint8 decimalsBase = feedBase.decimals();
```

```
35 uint256 SCALE_FACTOR = 18 + decimalsBase - decimalsQuote;
```

CVF-20 FIXED

- **Category** Suboptimal
- **Source** SRebalanceAdapter.sol

Description Performing individual call for setting every parameter could be inefficient. Also such approach could be inconvenient when several parameters should be set atomically.

Recommendation Implement a way to set several parameters at once.

Client Comment *Grouped setters into the 4 functions based on frequency and type of update. _isInvertedAssets (prev isInvertAssets) and _isNova(prev isUnicord) are now immutable, so setters are removed.*

```
51 function setRebalancePriceThreshold(uint256 _rebalancePriceThreshold
   ↪ ) external onlyOwner {

55 function setSqrtPriceAtLastRebalance(uint160
   ↪ _sqrtPriceAtLastRebalance) external onlyOwner {

59 function setOraclePriceAtLastRebalance(uint256
   ↪ _oraclePriceAtLastRebalance) external onlyOwner {

63 function setTimeAtLastRebalance(uint256 _timeAtLastRebalance)
   ↪ external onlyOwner {

67 function setRebalanceTimeThreshold(uint256 _rebalanceTimeThreshold)
   ↪ external onlyOwner {

71 function setWeight(uint256 _weight) external onlyOwner {

75 function setLongLeverage(uint256 _longLeverage) external onlyOwner {

79 function setShortLeverage(uint256 _shortLeverage) external onlyOwner
   ↪  {

83 function setMaxDeviationLong(uint256 _maxDeviationLong) external
   ↪ onlyOwner {

87 function setMaxDeviationShort(uint256 _maxDeviationShort) external
   ↪ onlyOwner {

91 function setIsInvertAssets(bool _isInvertAssets) external onlyOwner
   ↪ {

95 function setIsUnicord(bool _isUnicord) external onlyOwner {
```

(... 99)

CVF-21 FIXED

- **Category** Unclear behavior
- **Source** SRebalanceAdapter.sol

Description There are no range checks for the arguments.

Recommendation Implement proper checks.

Client Comment *Not all of them need checks, but some are added.*

```
51 function setRebalancePriceThreshold(uint256 _rebalancePriceThreshold
   ↪ ) external onlyOwner {
```

```
55 function setSqrtPriceAtLastRebalance(uint160
   ↪ _sqrtPriceAtLastRebalance) external onlyOwner {
```

```
59 function setOraclePriceAtLastRebalance(uint256
   ↪ _oraclePriceAtLastRebalance) external onlyOwner {
```

```
63 function setTimeAtLastRebalance(uint256 _timeAtLastRebalance)
   ↪ external onlyOwner {
```

```
67 function setRebalanceTimeThreshold(uint256 _rebalanceTimeThreshold)
   ↪ external onlyOwner {
```

```
71 function setWeight(uint256 _weight) external onlyOwner {
```

```
75 function setLongLeverage(uint256 _longLeverage) external onlyOwner {
```

```
79 function setShortLeverage(uint256 _shortLeverage) external onlyOwner
   ↪  {
```

```
83 function setMaxDeviationLong(uint256 _maxDeviationLong) external
   ↪ onlyOwner {
```

(... 87)

CVF-22 FIXED

- **Category** Unclear behavior
- **Source** SRebalanceAdapter.sol

Description It would make more sense to use cachedRatio and 1e18/cachedRatio instead of cachedRatio - 1e18 and1e18 - cachedRatio.

```
115 uint256 priceThreshold = oraclePrice > oraclePriceAtLastRebalance ?
    ↪ cachedRatio - 1e18 : 1e18 - cachedRatio;
```

CVF-23 FIXED

- **Category** Suboptimal
- **Source** SRebalanceAdapter.sol

Description The value “token.balanceOf(this)” is obtained twice.

Recommendation Obtain once and reuse.

Client Comment *The same applies to lines 197-198, 137-138, 141-142.*

```
184 if (amount > IERC20(token).balanceOf(address(this))) {
        swapAdapter.swapExactOutput(otherToken(token), token, amount -
            ↪ IERC20(token).balanceOf(address(this)));
```

CVF-24 FIXED

- **Category** Suboptimal
- **Source** SRebalanceAdapter.sol

Description Here the same contract is called multiple times, which is inefficient.

Recommendation Merge several calls into a single one.

Client Comment *The same applies to lines 261-264, 299-303; ALM 98-99, 130-134, 147-151. PositionManager 50-51, UnicordPositionManager 37-38, 46-47.*

```
207 if (deltaCL > 0) lendingAdapter.addCollateralLong(uint256(deltaCL));
    if (deltaCS > 0) lendingAdapter.addCollateralShort(uint256(deltaCS))
        ↪ ;
    if (deltaDL < 0) lendingAdapter.repayLong(uint256(-deltaDL));
210 if (deltaDS < 0) lendingAdapter.repayShort(uint256(-deltaDS));
    if (deltaCL < 0) lendingAdapter.removeCollateralLong(uint256(-
        ↪ deltaCL));
    if (deltaCS < 0) lendingAdapter.removeCollateralShort(uint256(-
        ↪ deltaCS));
    if (deltaDL > 0) lendingAdapter.borrowLong(uint256(deltaDL));
    if (deltaDS > 0) lendingAdapter.borrowShort(uint256(deltaDS));
```

CVF-25 FIXED

- **Category** Overflow/Underflow
- **Source** SRebalanceAdapter.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculation overflows.

Recommendation Use the “mulDiv” function.

Client Comment *237, 243 are not a real “phantom overflow”.*

```
234 targetCL = alm.TVL().mul(weight).mul(longLeverage).div(price);
```

```
237 targetDL = targetCL.mul(price).mul(1e18 - uint256(1e18).div(
    ↪ longLeverage));
    targetDS = targetCS.div(price).mul(1e18 - uint256(1e18).div(
    ↪ shortLeverage));
```

```
243 targetDL = targetCL.mul(price).mul(1e18 - uint256(1e18).div(
    ↪ longLeverage));
    targetDS = targetCS.div(price).mul(1e18 - uint256(1e18).div(
    ↪ shortLeverage));
```

CVF-26 FIXED

- **Category** Suboptimal
- **Source** SRebalanceAdapter.sol

Description The value “alm.TVL()” is obtained several times.

Recommendation Obtain once and reuse.

```
234 targetCL = alm.TVL().mul(weight).mul(longLeverage).div(price);
    targetCS = alm.TVL().mul(1e18 - weight).mul(shortLeverage);
```

```
240 targetCL = alm.TVL().mul(weight).mul(longLeverage);
    targetCS = alm.TVL().mul(1e18 - weight).mul(shortLeverage).mul(price
    ↪ );
```

CVF-27 FIXED

- **Category** Suboptimal
- **Source** SRebalanceAdapter.sol

Description The value “alm.isInvertedPool()” is calculated twice.

Recommendation Calculate once and reuse.

Client Comment *Now isInvertedPool is immutable.*

```
285 alm.isInvertedPool(),
```

```
290 alm.isInvertedPool(),
```

CVF-28 FIXED

- **Category** Unclear behavior
- **Source** ALM.sol

Description There is no way to the caller to specify the minimum shares to be minted.

Recommendation Consider adding a “minShares” argument.

```
73 _mint(to, _shares);
```

CVF-29 FIXED

- **Category** Suboptimal
- **Source** ALM.sol

Description Here the same contract is called several times, which is inefficient.

Recommendation Merge several calls into a single one.

Client Comment *The same 333-336.*

```
88 lendingAdapter.getCollateralLong(),
   lendingAdapter.getCollateralShort(),
90 lendingAdapter.getBorrowedLong(),
   lendingAdapter.getBorrowedShort()
```

9 Recommendations

CVF-30 FIXED

- **Category** Procedural
- **Source** ALMMathLib.sol

Description Consider specifying as “^0.8.0” unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-31 INFO

- **Category** Procedural
- **Source** ALMMathLib.sol

Description We didn’t review this file.

```
5 import {PRBMathUD60x18} from "@prb-math/PRBMathUD60x18.sol";
```

CVF-32 FIXED

- **Category** Readability
- **Source** ALMMathLib.sol

Description Brackets around multiplication are redundant.

Recommendation Remove redundant brackets.

Client Comment *The functions were removed and changed to v4 native.*

```
22 uint160 sqrtPriceDeltaX96 = SafeCast.toUint160((amount1 * Q96) /
   ↪ liquidity);
```

```
31 uint160 sqrtPriceDeltaX96 = SafeCast.toUint160((amount1 * Q96) /
   ↪ liquidity);
```

CVF-33 FIXED

- **Category** Suboptimal
- **Source** ALMMathLib.sol

Description Conversions of “sqrtPriceCurrentX96” to “uint256” are redundant, as compiler would do these conversions anyway.

Recommendation Remove redundant conversions.

Client Comment *The functions were removed and changed to v4 native.*

```
42 uint256(liquidity).mul(uint256(sqrtPriceCurrentX96)).div(
       uint256(liquidity) - amount0.mul(uint256(sqrtPriceCurrentX96)).
          ↪ div(Q96)
```

```
55 uint256(liquidity).mul(uint256(sqrtPriceCurrentX96)).div(
       uint256(liquidity) + amount0.mul(uint256(sqrtPriceCurrentX96)).
          ↪ div(Q96)
```

CVF-34 FIXED

- **Category** Suboptimal
- **Source** ALMMathLib.sol

Description Right shift would be more efficient, than division by a power of two.

Client Comment *The functions were removed and changed to v4 native.*

```
43             uint256(liquidity) - amount0.mul(uint256(
                  ↪ sqrtPriceCurrentX96)).div(Q96)
```

```
56             uint256(liquidity) + amount0.mul(uint256(
                  ↪ sqrtPriceCurrentX96)).div(Q96)
```

```
144 return uint256(sqrtPriceX96).pow(2 * WAD).mul(WAD * WAD).div(Q192);
```

CVF-35 FIXED

- **Category** Suboptimal
- **Source** ALMMathLib.sol

Description Conversions of “liquidity” to “uint256” are redundant here, as compiler would do these conversions anyway.

Recommendation Remove redundant conversions.

Client Comment *The functions were removed and changed to v4 native.*

```
43 uint256(liquidity) - amount0.mul(uint256(sqrtPriceCurrentX96)).div(
       ↪ Q96)
```

```
56 uint256(liquidity) + amount0.mul(uint256(sqrtPriceCurrentX96)).div(
       ↪ Q96)
```

CVF-36 FIXED

- **Category** Readability
- **Source** ALMMathLib.sol

Description Outer brackets are redundant.

Recommendation Remove redundant brackets.

Client Comment *The functions were removed and changed to v4 native.*

```
66 return (LiquidityAmounts.getAmount0ForLiquidity(sqrtPriceNextX96,
       ↪ sqrtPriceCurrentX96, liquidity));
```

```
74 return (LiquidityAmounts.getAmount1ForLiquidity(sqrtPriceNextX96,
       ↪ sqrtPriceCurrentX96, liquidity));
```

CVF-37 FIXED

- **Category** Bad naming
- **Source** ALMMathLib.sol

Description These names are confusing.

Recommendation Provide detailed explanations of the function logic in a documentation comment.

```
83 uint256 EH,
   uint256 UH,
   uint256 CL,
   uint256 DS,
   uint256 CS,
   uint256 DL,
```

CVF-38 FIXED

- **Category** Bad naming
- **Source** ALMMathLib.sol

Description The function names are confusing.

Recommendation Provide detailed explanation of the function logic in documentation comments.

Client Comment *The functions were removed and changed to v4 native.*

```
101 function getVLP(
```

```
115 function getL(uint256 VLP, uint256 price, uint256 priceUpper,
       ↪ uint256 priceLower) internal pure returns (uint256) {
```

CVF-39 FIXED

- **Category** Bad datatype
- **Source** ALMMathLib.sol

Description The constant values used here should be named constants with descriptive names.

Client Comment *The functions were removed and changed to v4 native.*

```
116 return VLP.div((2 * WAD).mul(price.sqrt()) - priceLower.sqrt() -
       ↪ price.div(priceUpper.sqrt())) / 1e6;
```

```
135         ((SafeCast.toInt256(PRBMathUD60x18.ln(price * WAD)) - int256
              ↪ (41446531673892820000))) / 99995000333297
```

CVF-40 FIXED

- **Category** Procedural
- **Source** TokenWrapperLib.sol

Description Consider specifying as “^0.8.0” unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.20;
```

CVF-41 FIXED

- **Category** Bad naming
- **Source** TokenWrapperLib.sol

Description This constant name is confusing, as the name WAD is commonly used for the value 10^18.

Recommendation Use a different name.

Client Comment *Will use WAD_DECIMALS instead.*

```
5 uint8 internal constant WAD = 18;
```

CVF-42 FIXED

- **Category** Bad naming
- **Source** TokenWrapperLib.sol

Description The name “tokenWAD” is confusing.

Recommendation Use more conventional “tokenDecimals”.

Client Comment *Functions were removed.*

```
7  function wrap(uint256 a, uint8 tokenWAD) internal pure returns (
       ↪ uint256) {
13 function unwrap(uint256 a, uint8 tokenWAD) internal pure returns (
       ↪ uint256) {
```

CVF-43 INFO

- **Category** Procedural
- **Source** CurrencySettlerSafe.sol

Description Consider specifying as "^0.8.0" unless there is something special regarding this particular version.

Client Comment *This was a copy of OpenZeppelin's Currency Settler with SafeERC20 modifications, so better not to change it. Now my PR with those modifications was approved, so it's the exact copy.*

```
2 pragma solidity ^0.8.20;
```

CVF-44 FIXED

- **Category** Procedural
- **Source** UniswapV3SwapAdapter.sol

Description Consider specifying as "^0.8.0" unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-45 FIXED

- **Category** Readability
- **Source** UniswapV3SwapAdapter.sol

Description These imports look like referring to third-party modules, while actually they refer to the same code base.

Recommendation Use relative paths.

```
5 import {Base} from "@src/core/base/Base.sol";
```

```
8 import {ISwapAdapter} from "@src/interfaces/ISwapAdapter.sol";
  import {ISwapRouter} from "@src/interfaces/swapAdapters/ISwapRouter.
     ↪ sol";
10 import {IUniswapV3Pool} from "@src/interfaces/swapAdapters/
     ↪ IUniswapV3Pool.sol";
```

CVF-46 FIXED

- **Category** Bad datatype
- **Source** UniswapV3SwapAdapter.sol

Description The type for this variable should be “IUniswapV3Pool”.

```
19 address public targetPool;
```

CVF-47 FIXED

- **Category** Readability
- **Source** UniswapV3SwapAdapter.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

Client Comment *It is not empty anymore.*

```
22 constructor() Base(msg.sender) {}
```

CVF-48 FIXED

- **Category** Unclear behavior
- **Source** UniswapV3SwapAdapter.sol

Description This function should emit some event.

```
29 function setTargetPool(address _targetPool) external onlyOwner {
```

CVF-49 FIXED

- **Category** Bad datatype
- **Source** UniswapV3SwapAdapter.sol

Description The type for the “tokenIn” and “tokenOut” arguments should be “IERC20”.

Client Comment *Fixed also for swapExactOutput.*

```
33 function swapExactInput(address tokenIn, address tokenOut, uint256
   ↪ amountIn) external onlyModule returns (uint256) {
```

CVF-50 FIXED

- **Category** Procedural
- **Source** PositionManager.sol

Description Consider specifying as “^0.8.0” unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-51 INFO

- **Category** Procedural
- **Source** PositionManager.sol

Description We didn’t review this file.

```
5 import {PRBMathUD60x18} from "@prb-math/PRBMathUD60x18.sol";
```

CVF-52 FIXED

- **Category** Readability
- **Source** PositionManager.sol

Description These imports look like referring to third-party modules, while actually they refer to the same code base.

Recommendation Use relative paths.

```
9 import {Base} from "@src/core/base/Base.sol";
```

```
15 import {IPositionManager} from "@src/interfaces/IPositionManager.sol
   ↪ ";
```

CVF-53 FIXED

- **Category** Procedural
- **Source** PositionManager.sol

Description Imports from the same code base are intermixed with external imports.

Recommendation Group external imports together.

```
12 import {SafeERC20} from "@openzeppelin/contracts/token/ERC20/utils/
   ↪ SafeERC20.sol";
```

```
16 import {IERC20} from "@openzeppelin/contracts/token/ERC20/IERC20.sol
   ↪ ";
```

CVF-54 FIXED

- **Category** Readability
- **Source** PositionManager.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

```
31 constructor() Base(msg.sender) {}
```

CVF-55 FIXED

- **Category** Unclear behavior
- **Source** PositionManager.sol

Description These functions should emit some events.

Client Comment *setFees is moved to Hook contract.*

```
33 function setKParams(uint256 _k1, uint256 _k2) external onlyOwner {
```

```
38 function setFees(uint256 _fees) external onlyOwner {
```

CVF-56 FIXED

- **Category** Procedural
- **Source** UnicordPositionManager.sol

Description Consider specifying as “^0.8.0” unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-57 INFO

- **Category** Procedural
- **Source** UnicordPositionManager.sol

Description We didn’t review this file.

```
5 import {PRBMathUD60x18} from "@prb-math/PRBMathUD60x18.sol";
```

CVF-58 FIXED

- **Category** Readability
- **Source** UnicordPositionManager.sol

Description These imports look like referring to third-party modules, while actually they refer to the same code base.

Recommendation Use relative paths.

```
6  import {TokenWrapperLib} from "@src/libraries/TokenWrapperLib.sol";

9  import {Base} from "@src/core/base/Base.sol";

15 import {IPositionManager} from "@src/interfaces/IPositionManager.sol
   ↪ ";
```

CVF-59 FIXED

- **Category** Procedural
- **Source** UnicordPositionManager.sol

Description Imports from the same codebase are intermixed with external imports.

Recommendation Group external imports together.

```
12 import {SafeERC20} from "@openzeppelin/contracts/token/ERC20/utils/
   ↪ SafeERC20.sol";

16 import {IERC20} from "@openzeppelin/contracts/token/ERC20/IERC20.sol
   ↪ ";
```

CVF-60 FIXED

- **Category** Readability
- **Source** UnicordPositionManager.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

```
28 constructor() Base(msg.sender) {}
```

CVF-61 INFO

- **Category** Suboptimal
- **Source** UnicordPositionManager.sol

Description This function is redundant, as the “fees” variable is already public.

Recommendation Either remove this function or make the “feel” variable non-public.

Client Comment *It was intended to have a possibility to update PositionManager to use dynamic fees in the future. Thus, the separate getter with two variables was necessary. Now the fee logic is uniswap native, so the function is removed.*

```
52 function getSwapFees(bool, int256) external view returns (uint256) {
```

CVF-62 FIXED

- **Category** Procedural
- **Source** MorphoLendingAdapter.sol

Description Consider specifying as “^0.8.0” unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-63 FIXED

- **Category** Procedural
- **Source** MorphoLendingAdapter.sol

Description We didn’t review these files.

```
5 import {IMorpho, Id, Position} from "@morpho-blue/interfaces/IMorpho
    ↪ .sol";
  import {MorphoBalancesLib} from "@morpho-blue/libraries/periphery/
    ↪ MorphoBalancesLib.sol";
```

CVF-64 FIXED

- **Category** Readability
- **Source** MorphoLendingAdapter.sol

Description These imports look like referring to third-party modules, while actually they refer to the same code base.

Recommendation Use relative paths.

```
9  import {TokenWrapperLib} from "@src/libraries/TokenWrapperLib.sol";

13 import {Base} from "@src/core/base/Base.sol";

18 import {ILendingAdapter, IFlashLoanReceiver} from "@src/interfaces/
       ↪ ILendingAdapter.sol";
```

CVF-65 FIXED

- **Category** Procedural
- **Source** MorphoLendingAdapter.sol

Description Imports from the same codebase are intermixed with external imports.

Recommendation Group external imports together.

```
10 import {SafeERC20} from "@openzeppelin/contracts/token/ERC20/utils/
       ↪ SafeERC20.sol";

16 import {IERC20} from "@openzeppelin/contracts/token/ERC20/IERC20.sol
       ↪ ";
   import {IERC4626} from "@openzeppelin/contracts/interfaces/IERC4626.
       ↪ sol";
```

CVF-66 FIXED

- **Category** Bad datatype
- **Source** MorphoLendingAdapter.sol

Description The type for the “_earnBase” and “_earnQuote” arguments should be “IERC4626”.

```
34 constructor(Id _longMId, Id _shortMId, address _earnBase, address
       ↪ _earnQuote) Base(msg.sender) {
```

CVF-67 FIXED

- **Category** Unclear behavior
- **Source** MorphoLendingAdapter.sol

Description This transfer should be performed in the “loanType == 2” section for code consistency.

```
97 IERC20(asset0).safeTransfer(sender, amount0);
```

CVF-68 FIXED

- **Category** Procedural
- **Source** EulerLendingAdapter.sol

Description Consider specifying as “^0.8.0” unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-69 FIXED

- **Category** Readability
- **Source** EulerLendingAdapter.sol

Description These imports look like referring to third-party modules, while actually they refer to the same code base.

Recommendation Use relative paths.

```
5  import {IEulerVault} from "@src/interfaces/lendingAdapters/
       ↪ IEulerVault.sol";
   import {IEVC} from "@src/interfaces/lendingAdapters/IEVC.sol";

9  import {TokenWrapperLib} from "@src/libraries/TokenWrapperLib.sol";

13 import {Base} from "@src/core/base/Base.sol";

17 import {ILendingAdapter, IFlashLoanReceiver} from "@src/interfaces/
       ↪ ILendingAdapter.sol";
```

CVF-70 FIXED

- **Category** Procedural
- **Source** EulerLendingAdapter.sol

Description Imports from the same code base are intermixed with external imports.

Recommendation Group external imports together.

```
10 import {SafeERC20} from "@openzeppelin/contracts/token/ERC20/utils/
      ↪ SafeERC20.sol";

16 import {IERC20} from "@openzeppelin/contracts/token/ERC20/IERC20.sol
      ↪ ";
```

CVF-71 FIXED

- **Category** Bad datatype
- **Source** EulerLendingAdapter.sol

Description The type for the arguments should be “IEulerVault”.

```
32 constructor(address _vault0, address _vault1, address _flVault0,
      ↪ address _flVault1) Base(msg.sender) {
```

CVF-72 INFO

- **Category** Bad datatype
- **Source** EulerLendingAdapter.sol

Description The type conversions are redundant, as the values already have type “address”.

Recommendation Remove redundant conversions.

Client Comment *After fixing issue 71, these are now necessary.*

```
38 evc.enableController(subAccount0, address(vault0));
   evc.enableCollateral(subAccount0, address(vault1));
40 evc.enableController(subAccount1, address(vault1));
   evc.enableCollateral(subAccount1, address(vault0));
```

CVF-73 FIXED

- **Category** Suboptimal
- **Source** EulerLendingAdapter.sol

Description This condition always holds as “accountId” has type “uint8”.

Recommendation Remove this check.

```
52 require(accountId < 256, "Invalid␣account␣ID");
```

CVF-74 FIXED

- **Category** Suboptimal
- **Source** EulerLendingAdapter.sol

Description Conversion to “uint8” is redundant, as the value is anyway encoded as 256 bits.

```
59 bytes memory _data = abi.encode(uint8(0), msg.sender, asset, amount,
   ↪  data);
```

```
70 bytes memory _data = abi.encode(uint8(2), msg.sender, asset0,
   ↪ amount0, asset1, amount1, data);
```

CVF-75 FIXED

- **Category** Unclear behavior
- **Source** EulerLendingAdapter.sol

Description This transfer should be performed in the “loanType == 2” section for code consistency.

```
113 IERC20(asset0).safeTransfer(sender, amount0);
```

CVF-76 FIXED

- **Category** Suboptimal
- **Source** EulerLendingAdapter.sol

Description Calling external vault every time is inefficient.

Recommendation Store the vault's tokens in this contract as immutable variables.

Client Comment *Refactored by splitting Lending Adapter into lending and flash loan adapters, and also moved duplicated code into the corresponding base contracts. Additionally, the asset ordering during a flash loan is now unchanged.*

```
124 if (flVault0.asset() == token) return flVault0;
    else if (flVault1.asset() == token) return flVault1;
```

CVF-77 FIXED

- **Category** Readability
- **Source** EulerLendingAdapter.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

Client Comment *Fixed, also now one function due to gas optimization.*

```
217 function syncLong() external {}
```

```
219 function syncShort() external {}
```

CVF-78 FIXED

- **Category** Procedural
- **Source** Base.sol

Description Consider specifying as "^0.8.0" unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-79 FIXED

- **Category** Readability
- **Source** Base.sol

Description These imports look like referring to third-party modules, while actually they refer to the same code base.

Recommendation Use relative paths.

```
5  import {IALM} from "@src/interfaces/IALM.sol";
   import {ILendingAdapter} from "@src/interfaces/ILendingAdapter.sol";
   import {IPositionManager} from "@src/interfaces/IPositionManager.sol
      ↪ ";
   import {IOracle} from "@src/interfaces/IOracle.sol";
   import {IRebalanceAdapter} from "@src/interfaces/IRebalanceAdapter.
      ↪ sol";
10 import {ISwapAdapter} from "@src/interfaces/ISwapAdapter.sol";
   import {IBase} from "@src/interfaces/IBase.sol";
```

CVF-80 FIXED

- **Category** Bad datatype
- **Source** Base.sol

Description The type for these variables should be “IERC20”.

Client Comment *Fixed, also now immutable.*

```
22 address public base;
   address public quote;
```

CVF-81 FIXED

- **Category** Unclear behavior
- **Source** Base.sol

Description In ERC-20 the “decimals” property is used by UI to render token amounts in human-readable way. Using this property in smart contracts is discouraged.

Recommendation Treat token amounts are integer values.

Client Comment *Fixed, now we don’t have decimals, only tokens decimals delta in the oracle.*

```
24 uint8 public bDec;
   uint8 public qDec;
```

CVF-82 INFO

- **Category** Unclear behavior
- **Source** Base.sol

Description Including the previous owner into these events is redundant, as the old owner could be derived from the previous event.

Recommendation Don't include the previous owner into these events.

Client Comment *I agree, but it was inspired by Openzeppelin's Ownable. And it's better to have similar event styles to simplify integration in the future.*

```
36 emit OwnershipTransferred(address(0), initialOwner);
```

```
86 emit OwnershipTransferred(oldOwner, owner);
```

CVF-83 FIXED

- **Category** Readability
- **Source** Base.sol

Description It is a good practice to put a comment into an empty block to explain, why the block is empty.

Client Comment *Function is removed.*

```
50 function _postSetTokens() internal virtual {}
```

CVF-84 FIXED

- **Category** Bad datatype
- **Source** Base.sol

Description The type for this argument should be "IALM".

```
53 address _alm,
```

CVF-85 FIXED

- **Category** Bad datatype
- **Source** Base.sol

Description The type for this argument should be "ILendingAdapter".

```
54 address _lendingAdapter,
```

CVF-86 FIXED

- **Category** Bad datatype
- **Source** Base.sol

Description The type for this argument should be “IPositionManager”.

```
55 address _positionManager,
```

CVF-87 FIXED

- **Category** Bad datatype
- **Source** Base.sol

Description The type for this argument should be “IOracle”.

```
56 address _oracle,
```

CVF-88 FIXED

- **Category** Bad datatype
- **Source** Base.sol

Description The type for this argument should be “IRebalanceAdapter”.

```
57 address _rebalanceAdapter,
```

CVF-89 FIXED

- **Category** Bad datatype
- **Source** Base.sol

Description The type for this argument should be “ISwapAdapter”.

```
58 address _swapAdapter
```

CVF-90 FIXED

- **Category** Unclear behavior
- **Source** Base.sol

Description In case moduleOld==moduleNew, the allowance shouldn’t be revoked either.

Recommendation Add “&& moduleOld != moduleNew” to the condition.

```
78 if (moduleOld != address(0) && moduleOld != address(this)) IERC20(
   ↪ token).forceApprove(moduleOld, 0);
```

CVF-91 FIXED

- **Category** Suboptimal
- **Source** Base.sol

Description This check is redundant, as it is anyway possible to pass a dead owner address.

Recommendation Remove the check.

```
83 if (newOwner == address(0)) revert OwnableInvalidOwner(address(0));
```

CVF-92 FIXED

- **Category** Suboptimal
- **Source** Base.sol

Description This assignment wouldn't be necessary if the event would be emitted before actually changing the stored owner address.

Recommendation Remove this assignment and emit event before assigning the "owner" variable.

```
84 address oldOwner = owner;
```

CVF-93 FIXED

- **Category** Suboptimal
- **Source** Base.sol

Description This event is emitted even if nothing actually changed.

```
86 emit OwnershipTransferred(oldOwner, owner);
```

CVF-94 FIXED

- **Category** Unclear behavior
- **Source** Base.sol

Description One of these events does include the “msg.sender” address, while others don’t.

Recommendation Include the “msg.sender’ address into all the events for consistency and to make easier investigating problems.

```
97      revert OwnableUnauthorizedAccount(msg.sender);
```

```
104 if (msg.sender != address(alm)) revert NotALM();
```

```
110 if (msg.sender != address(rebalanceAdapter)) revert
    ↪ NotRebalanceAdapter();
```

```
116 if (msg.sender != address(lendingAdapter)) revert NotLendingAdapter
    ↪ ();
```

```
128 ) revert NotModule();
```

CVF-95 FIXED

- **Category** Procedural
- **Source** BaseStrategyHook.sol

Description Consider specifying as “^0.8.0” unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-96 FIXED

- **Category** Readability
- **Source** BaseStrategyHook.sol

Description This particular “v4” import isn’t under the “v4 imports” comment.

Recommendation Move it next to the other “v4” imports.

```
4  // ** v4 imports
```

```
15 import {SafeCast} from "v4-core/libraries/SafeCast.sol";
```

CVF-97 FIXED

- **Category** Readability
- **Source** BaseStrategyHook.sol

Description These imports look like referring to third-party modules, while actually they refer to the same code base.

Recommendation Use relative paths.

```
13 import {ALMMathLib} from "@src/libraries/ALMMathLib.sol";
```

```
18 import {Base} from "@src/core/Base/Base.sol";
```

```
21 import {IALM} from "@src/interfaces/IALM.sol";
```

CVF-98 INFO

- **Category** Procedural
- **Source** BaseStrategyHook.sol

Description We didn't review this file.

```
14 import {PRBMathUD60x18} from "@prb-math/PRBMathUD60x18.sol";
```

CVF-99 FIXED

- **Category** Readability
- **Source** BaseStrategyHook.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

Client Comment *Not empty anymore.*

```
49 constructor(IPoolManager _poolManager) BaseHook(_poolManager) Base(
   ↪ msg.sender) {}
```

CVF-100 FIXED

- **Category** Unclear behavior
- **Source** BaseStrategyHook.sol

Description These functions should emit some events.

```
51  function setTickUpperDelta(int24 _tickUpperDelta) external onlyOwner
    ↪  {
55  function setTickLowerDelta(int24 _tickLowerDelta) external onlyOwner
    ↪  {
59  function setPaused(bool _paused) external onlyOwner {
63  function setShutdown(bool _shutdown) external onlyOwner {
67  function setIsInvertAssets(bool _isInvertAssets) external onlyOwner
    ↪ {
71  function setIsInvertedPool(bool _isInvertedPool) external onlyOwner
    ↪ {
75  function setSwapPriceThreshold(uint256 _swapPriceThreshold) external
    ↪  onlyOwner {
79  function setLiquidityOperator(address _liquidityOperator) external
    ↪ onlyOwner {
83  function setSwapOperator(address _swapOperator) external onlyOwner {
87  function setTVLCap(uint256 _tvlCap) external onlyOwner {
91  function setTreasury(address _treasury) external onlyOwner {
95  function setProtocolFee(uint256 _protocolFee) external onlyOwner {
133 function updateLiquidity(uint128 _liquidity) public
    ↪ onlyRebalanceAdapter {
137 function updateSqrtPrice(uint160 _sqrtPrice) public
    ↪ onlyRebalanceAdapter {
141 function _updateBoundaries() internal {
```

CVF-101 FIXED

- **Category** Suboptimal
- **Source** BaseStrategyHook.sol

Description Calling each function individually is inefficient in case several parameters should be set at once. Also, separate setter for each parameter could be inconvenient in case several parameters ought to be set atomically.

Recommendation Implement some way to set several parameters at once.

Client Comment *Grouped setters into the 4 functions based on frequency and type of update. _isInvertedAssets (prev isInvertAssets) and _isInvertedPool are now immutable, so setters are removed. Also, setPaused and setShutdown an emergency-level operations, so they should be called from a separate function. The Treasury setter is also separate because it will be called rarely. Other params are grouped under setProtocolParams.*

```
51 function setTickUpperDelta(int24 _tickUpperDelta) external onlyOwner
   ↪  {

55 function setTickLowerDelta(int24 _tickLowerDelta) external onlyOwner
   ↪  {

59 function setPaused(bool _paused) external onlyOwner {

63 function setShutdown(bool _shutdown) external onlyOwner {

67 function setIsInvertAssets(bool _isInvertAssets) external onlyOwner
   ↪ {

71 function setIsInvertedPool(bool _isInvertedPool) external onlyOwner
   ↪ {

75 function setSwapPriceThreshold(uint256 _swapPriceThreshold) external
   ↪  onlyOwner {

79 function setLiquidityOperator(address _liquidityOperator) external
   ↪ onlyOwner {

83 function setSwapOperator(address _swapOperator) external onlyOwner {

87 function setTVLCap(uint256 _tvlCap) external onlyOwner {

91 function setTreasury(address _treasury) external onlyOwner {

95 function setProtocolFee(uint256 _protocolFee) external onlyOwner {
```

CVF-102 FIXED

- **Category** Bad naming
- **Source** BaseStrategyHook.sol

Description These two functions are very similar.

Recommendation Refactor the code to reduce duplication.

Client Comment *Removed entirely, now relying on uniswap swap values calculation.*

```
151 function getZeroForOneDeltas(
```

```
189 function getOneForZeroDeltas(
```

CVF-103 FIXED

- **Category** Suboptimal
- **Source** BaseStrategyHook.sol

Description This line should be within an “unchecked” block to save gas and allow type(int256).minValue.

```
177 token0In = uint256(-amountSpecified);
```

```
215 token1In = uint256(-amountSpecified);
```

CVF-104 FIXED

- **Category** Procedural
- **Source** Oracle.sol

Description Consider specifying as “^0.8.0” unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-105 INFO

- **Category** Procedural
- **Source** Oracle.sol

Description We didn’t review these files.

```
5 import {PRBMath} from "@prb-math/PRBMath.sol";
```

```
9 import {AggregatorV3Interface} from "@chainlink/shared/interfaces/
    ↪ AggregatorV3Interface.sol";
```

CVF-106 FIXED

- **Category** Bad datatype
- **Source** Oracle.sol

Description The type for the "_feedQuote" and "_feedBase" arguments should be "AggregatorV3Interface".

```
17 constructor(address _feedQuote, address _feedBase, uint256
   ↪ _stalenessThresholdQ, uint256 _stalenessThresholdB) {
```

CVF-107 FIXED

- **Category** Suboptimal
- **Source** Oracle.sol

Description These lines will revert in case block.timestamp < stalenes threshold.

Recommendation Rewrite as: block.timestamp - updatedAtQuote <= stalenessthresholdQ block.timestamp - updatedAtBase <= stalenessThresholdB

```
32 require(updatedAtQuote >= block.timestamp - stalenessThresholdQ, "02
   ↪ ");
   require(updatedAtBase >= block.timestamp - stalenessThresholdB, "03"
   ↪ );
```

CVF-108 FIXED

- **Category** Suboptimal
- **Source** Oracle.sol

Description These checks should be performed earlier. Right after obtaining prices.

```
32 require(updatedAtQuote >= block.timestamp - stalenessThresholdQ, "02
   ↪ ");
   require(updatedAtBase >= block.timestamp - stalenessThresholdB, "03"
   ↪ );
```

CVF-109 FIXED

- **Category** Bad naming
- **Source** Oracle.sol

Description UPPER_CASE names are commonly used for constants.

Recommendation Use camelCase here.

```
35 uint256 SCALE_FACTOR = 18 + decimalsBase - decimalsQuote;
```

CVF-110 FIXED

- **Category** Procedural
- **Source** SRebalanceAdapter.sol

Description Consider specifying as “^0.8.0” unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-111 FIXED

- **Category** Readability
- **Source** SRebalanceAdapter.sol

Description These imports look like referring to third-party modules, while actually they refer to the same code base.

Recommendation Use relative paths.

```
5 import {ALMMathLib} from "@src/libraries/ALMMathLib.sol";
```

```
7 import {TokenWrapperLib} from "@src/libraries/TokenWrapperLib.sol";
```

```
11 import {Base} from "@src/core/base/Base.sol";
```

```
14 import {IRebalanceAdapter} from "@src/interfaces/IRebalanceAdapter.
   ↪ sol";
```

CVF-112 INFO

- **Category** Procedural
- **Source** SRebalanceAdapter.sol

Description We didn’t review this file.

```
6 import {PRBMathUD60x18} from "@prb-math/PRBMathUD60x18.sol";
```

CVF-113 FIXED

- **Category** Procedural
- **Source** SRebalanceAdapter.sol

Description Imports from the same codebase are intermixed with external imports.

Recommendation Group external imports together.

```
6  import {PRBMathUD60x18} from "@prb-math/PRBMathUD60x18.sol";
8  import {SafeCast} from "v4-core/libraries/SafeCast.sol";
15 import {IERC20} from "@openzeppelin/token/ERC20/IERC20.sol";
```

CVF-114 FIXED

- **Category** Bad naming
- **Source** SRebalanceAdapter.sol

Description The error name doesn't look like an error, but rather like a normal situation.

Recommendation Use a different name.

```
18 error NoRebalanceNeeded();
```

CVF-115 FIXED

- **Category** Readability
- **Source** SRebalanceAdapter.sol

Description The number format for these fields is unclear.

Recommendation Explain in a documentation comment.

```
24 uint256 slippage,
26 uint256 oraclePriceAtRebalance,
```

CVF-116 FIXED

- **Category** Readability
- **Source** SRebalanceAdapter.sol

Description The number format for these variables is unclear.

Recommendation Explain in a documentation comment.

```
40 uint256 public weight;
   uint256 public longLeverage;
   uint256 public shortLeverage;
```

CVF-117 INFO

- **Category** Readability
- **Source** SRebalanceAdapter.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

Client Comment *Not empty due to moving some variables to immutable.*

```
49 constructor() Base(msg.sender) {}
```

CVF-118 FIXED

- **Category** Unclear behavior
- **Source** SRebalanceAdapter.sol

Description These functions should emit some events.

```
51 function setRebalancePriceThreshold(uint256 _rebalancePriceThreshold
      ↪ ) external onlyOwner {
```

```
55 function setSqrtPriceAtLastRebalance(uint160
      ↪ _sqrtPriceAtLastRebalance) external onlyOwner {
```

```
59 function setOraclePriceAtLastRebalance(uint256
      ↪ _oraclePriceAtLastRebalance) external onlyOwner {
```

```
63 function setTimeAtLastRebalance(uint256 _timeAtLastRebalance)
      ↪ external onlyOwner {
```

```
67 function setRebalanceTimeThreshold(uint256 _rebalanceTimeThreshold)
      ↪ external onlyOwner {
```

```
71 function setWeight(uint256 _weight) external onlyOwner {

75 function setLongLeverage(uint256 _longLeverage) external onlyOwner {

79 function setShortLeverage(uint256 _shortLeverage) external onlyOwner
   ↪  {

83 function setMaxDeviationLong(uint256 _maxDeviationLong) external
   ↪ onlyOwner {

87 function setMaxDeviationShort(uint256 _maxDeviationShort) external
   ↪ onlyOwner {

91 function setIsInvertAssets(bool _isInvertAssets) external onlyOwner
   ↪ {

95 function setIsUnicord(bool _isUnicord) external onlyOwner {

99 function setRebalanceOperator(address _rebalanceOperator) external
   ↪ onlyOwner {
```

CVF-119 FIXED

- **Category** Readability
- **Source** SRebalanceAdapter.sol

Description The semantics of the returned values is unclear.

Recommendation Give descriptive names to the returned values and/or explain in a documentation comment.

```
105 function isRebalanceNeeded() public view returns (bool, uint256,
    ↪ uint256) {

112 function isPriceRebalance() public view returns (bool, uint256) {

120 function isTimeRebalance() public view returns (bool, uint256) {
```

CVF-120 FIXED

- **Category** Unclear behavior
- **Source** SRebalanceAdapter.sol

Description This should be checked via a modifier.

```
126 if (msg.sender != rebalanceOperator) revert NotRebalanceOperator();
```

CVF-121 FIXED

- **Category** Suboptimal
- **Source** SRebalanceAdapter.sol

Description The values “baseBalanceUnwr()” and “quoteBalanceUnwr()” are calculated several times.

Recommendation Calculate once and reuse.

```
137 if (baseBalanceUnwr() != 0) lendingAdapter.addCollateralShort((
    ↪ baseBalanceUnwr()).wrap(bDec));
if (quoteBalanceUnwr() != 0) lendingAdapter.addCollateralLong((
    ↪ quoteBalanceUnwr()).wrap(qDec));
```

```
141 if (baseBalanceUnwr() != 0) lendingAdapter.repayLong((
    ↪ baseBalanceUnwr()).wrap(bDec));
if (quoteBalanceUnwr() != 0) lendingAdapter.repayShort((
    ↪ quoteBalanceUnwr()).wrap(qDec));
```

CVF-122 FIXED

- **Category** Bad datatype
- **Source** SRebalanceAdapter.sol

Description The type for this argument should be “IERC20”.

```
179 address token,
```

CVF-123 FIXED

- **Category** Bad datatype
- **Source** SRebalanceAdapter.sol

Description The type for these arguments should be “IERC20”.

190
```
address base,
```

192
```
address quote,
```

CVF-124 FIXED

- **Category** Bad naming
- **Source** SRebalanceAdapter.sol

Description The function name is confusing. It looks like a getter, but doesn’t return any value.

202
```
function _positionManagement(bytes calldata data) internal {
```

CVF-125 FIXED

- **Category** Suboptimal
- **Source** SRebalanceAdapter.sol

Description These lines are the same in both branches.

Recommendation Put them after the conditional statement to reduce code duplication.

237
```
targetDL = targetCL.mul(price).mul(1e18 - uint256(1e18).div(
   ↪ longLeverage));
targetDS = targetCS.div(price).mul(1e18 - uint256(1e18).div(
   ↪ shortLeverage));
```

243
```
targetDL = targetCL.mul(price).mul(1e18 - uint256(1e18).div(
   ↪ longLeverage));
targetDS = targetCS.div(price).mul(1e18 - uint256(1e18).div(
   ↪ shortLeverage));
```

CVF-126 FIXED

- **Category** Suboptimal
- **Source** SRebalanceAdapter.sol

Description The expression "ALMMathLib.getVLP(alm.TVL(), weight, longLeverage, shortLeverage)" is coded twice.

Recommendation Calculate this expression at one place, before the conditional statement, to reduce code duplication.

Client Comment *Now using uniswap formulas here. Also function is moved to BaseStrategyHook.*

```
277 if (isInvertAssets) VLP = ALMMathLib.getVLP(alm.TVL(), weight,
       ↪ longLeverage, shortLeverage);
    else VLP = ALMMathLib.getVLP(alm.TVL(), weight, longLeverage,
       ↪ shortLeverage).mul(oracle.price());
```

CVF-127 FIXED

- **Category** Suboptimal
- **Source** SRebalanceAdapter.sol

Description This expression is calculated twice.

Recommendation Calculate once and reuse.

Client Comment *Now using uniswap formulas here. Also function is moved to BaseStrategyHook.*

```
286 uint8(ALMMathLib.absSub(bDec, qDec))
```

```
291 uint8(ALMMathLib.absSub(bDec, qDec))
```

CVF-128 FIXED

- **Category** Procedural
- **Source** ALM.sol

Description Consider specifying as "^0.8.0" unless there is something special regarding this particular version.

```
2 pragma solidity ^0.8.25;
```

CVF-129 FIXED

- **Category** Readability
- **Source** ALM.sol

Description These imports look like referring to third-party modules, while actually they refer to the same code base.

Recommendation Use relative paths.

```
13 import {ALMMathLib} from "@src/libraries/ALMMathLib.sol";

15 import {TokenWrapperLib} from "@src/libraries/TokenWrapperLib.sol";

20 import {BaseStrategyHook} from "@src/core/base/BaseStrategyHook.sol"
   ↪ ;
```

CVF-130 FIXED

- **Category** Procedural
- **Source** ALM.sol

Description Imports from the same code base are intermixed with external imports.

Recommendation Group external imports together.

```
14 import {SafeERC20} from "@openzeppelin/contracts/token/ERC20/utils/
   ↪ SafeERC20.sol";

16 import {PRBMathUD60x18} from "@prb-math/PRBMathUD60x18.sol";
   import {SafeCast} from "v4-core/libraries/SafeCast.sol";

21 import {ERC20} from "@openzeppelin/token/ERC20/ERC20.sol";

24 import {IERC20} from "@openzeppelin/token/ERC20/IERC20.sol";
```

CVF-131 INFO

- **Category** Procedural
- **Source** ALM.sol

Description We didn't review this file.

```
16 import {PRBMathUD60x18} from "@prb-math/PRBMathUD60x18.sol";
```

CVF-132 INFO

- **Category** Readability
- **Source** ALM.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

Client Comment *Not empty because some params are set here.*

```
40 ) BaseStrategyHook(manager) ERC20(name, symbol) {}
```

CVF-133 FIXED

- **Category** Readability
- **Source** ALM.sol

Description The semantics of the returned values is unclear.

Recommendation Give descriptive names to the returned values and/or explain in a documentation comment.

Client Comment *I also reduced the number of return variables because a function always takes in an amount = amountIn.*

```
56 function deposit(address to, uint256 amountIn) external notPaused
   ↪ notShutdown returns (uint256, uint256) {
```

CVF-134 FIXED

- **Category** Suboptimal
- **Source** ALM.sol

Description This check is redundant, as it will anyway be performed inside “_burn”.

```
81 if (balanceOf(msg.sender) < sharesOut) revert
   ↪ NotEnoughSharesToWithdraw();
```

CVF-135 FIXED

- **Category** Bad datatype
- **Source** ALM.sol

Description The type for these arguments should be “IERC20”.

```
122 address base,
```

```
124 address quote,
```

```
141 address token,
```

CVF-136 FIXED

- **Category** Bad datatype
- **Source** ALM.sol

Description The type for the “token” argument should be “IEC20”.

```
162 function _ensureEnoughBalance(uint256 balance, address token)
    ↪ internal {
```

CVF-137 FIXED

- **Category** Suboptimal
- **Source** ALM.sol

Description The expression “fee.mul(protocolFee)” is calculated several times.

Recommendation Calculate once and reuse.

```
216 accumulatedFeeB += fee.mul(protocolFee); //cut protocol fee from the
    ↪  calculated swap fee
```

```
218     (token0In - fee.mul(protocolFee)).wrap(bDec),
```

```
222 accumulatedFeeQ += fee.mul(protocolFee);
```

```
225     (token0In - fee.mul(protocolFee)).wrap(qDec)
```

CVF-138 FIXED

- **Category** Suboptimal
- **Source** ALM.sol

Description The value “IERC20(base).balanceOf(address(this)) - accumulatedFeeB” is calculated in both branches.

Recommendation Calculate in one place, before the ternary operator, to reduce code duplication.

```
315 ? (IERC20(base).balanceOf(address(this)) - accumulatedFeeB).wrap(
    ↪ bDec)
: IERC20(base).balanceOf(address(this)) - accumulatedFeeB;
```

CVF-139 FIXED

- **Category** Suboptimal
- **Source** ALM.sol

Description The expression “IERC20(quote).balanceOf(address(this)) - accumulated-FeeQ” is calculated in both branches.

Recommendation Calculate in one place, before the ternary operator, to reduce code duplication.

322
```
? (IERC20(quote).balanceOf(address(this)) - accumulatedFeeQ).wrap(
    ↪ qDec)
: IERC20(quote).balanceOf(address(this)) - accumulatedFeeQ;
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting