Report

v. 2.0

Customer

Trader Joe

Smart Contract Audit

Trader Joe

4th October, 2022

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	4.10.22	A. Zveryanskaya	Initial Draft
0.2	04.10.22	A. Zveryanskaya	Minor revision
1.0	04.10.22	A. Zveryanskaya	Release
1.1	04.10.22	A. Zveryanskaya	bitStep > binStep
2.0	04.10.22	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting 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 Repository
- Fix Repository

Files:

/				
	LBFac-tory.sol	LBRouter.sol	LBPair.sol	LBFactory Helper.sol
	LBToken.sol			
interfaces/				
	IJoeFac-tory.sol	IJoePair.sol	ILBFac-tory.sol	ILBFactory Helper.sol
	ILBToken.sol	ILBPair.sol	ILB Router.sol	ILBFlashLoan Callback.sol
	IWAVAX.sol	IPending Ownable.sol		
libraries/				
	Bin Helper.sol	BitMath.sol	Buffer.sol	Math512Bits.sol
	Decoder.sol	Encoder.sol	Fee Helper.sol	Math128×128.sol
	Con stants.sol	Oracle.sol	Pending Ownable.sol	IReentrancy Guard.sol
	SafeCast.sol	Safe Math.sol	Samples.sol	SwapHelper.sol
	Token Helper.sol	TreeMath.sol		

4 Methodology

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

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best 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 and that 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 is 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 that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check 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.

5 Our findings

We found 1 critical, 26 major, and a few less important issues. All identified Critical and Major issues have been fixed.

Issues	Active	Fixed
Severity		
Critical	0	1
Major	6	20
Moderate	7	18
Minor	96	143

Fixed 182 out of 292 issues

6 Critical Issues

CVF-209. FIXED

- **Category** Flaw
- **Source** Buffer.sol

Description This formula is incorrect in case x + n - y is negative or exceed 256 bits.

Recommendation Should be: result = addmod (x, n - y % n, n);

```
32 result = (x + n - y) % n;
```

7 Major Issues

CVF-4. INFO

- **Category** Suboptimal
- **Source** LBRouter.sol

Description These functions perform multiple invocations of the "_LBPair" smart contract. External contract invocations are expensive.

Recommendation Consider moving the logic of these functions into the "LBPair" smart contract to avoid multiple external calls.

Client Comment *Contract would be too heavy if we did this*

106
```
function getSwapIn(
```

CVF-35. INFO

- **Category** Flaw
- **Source** LBPair.sol

Description There are no checks for these arguments.

Recommendation Consider adding appropriate checks, such as that neither token is zero, the tokens are not the same, and the token addresses go in the right order.

Client Comment *This is intentional, those checks are done by the factory. The goal is to reduce the contract size*

138
```
IERC20 _tokenX,
IERC20 _tokenY,
```

CVF-37. INFO

- **Category** Flaw
- **Source** LBPair.sol

Description There are no validity checks for this argument.

Recommendation Consider adding appropriate checks.

Client Comment *This is intentional, those checks are done by the factory. The goal is to reduce the contract size*

142
```
bytes32 _packedFeeParameters
```

CVF-48. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Description Flash loans use the same fee parameters as swaps. This makes the fees model less flexible.

Recommendation Consider using separate parameters for flash loans.

```
439 FeeHelper.FeesDistribution memory _feesX = _fp.getFeesDistribution(
    ↪ _fp.getFees(_amountXOut, 0));
440 FeeHelper.FeesDistribution memory _feesY = _fp.getFeesDistribution(
    ↪ _fp.getFees(_amountYOut, 0));
```

CVF-49. FIXED

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

Recommendation It would be more intuitive to make a call on the "to" address. Conflicts with the docstring.

```
449 ILBFlashLoanCallback(msg.sender).LBFlashLoanCallback(_feesX.total,
    ↪ _feesY.total, _data);
```

CVF-74. FIXED

- **Category** Flaw
- **Source** LBFactory.sol

Recommendation This event should also contain the "_binStep" parameter, as it is needed to identify a pair.

```
60 event LBPairCreated(IERC20 indexed tokenX, IERC20 indexed tokenY,
   ↪ ILBPair LBPair, uint256 pid);
```

CVF-88. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Description This check won't help in case "_nbPresets" is zero.

Recommendation Consider returning early in such a case.

```
179 if (++_index == _nbPresets) break;
```

CVF-89. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Description This check won't help in case "_nbAvailable" is zero.

Recommendation Consider returning early in such a case.

```
213 if (++_index == _nbAvailable) break;
```

CVF-97. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Description In case the pair doesn't exist, this line will call a zero address which effectively will do nothing.

Recommendation Consider reverting on a non-existing pair.

```
428 _LBPair.setFeesParameters(_packedFeeParameters);
```

CVF-98. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Description This event is emitted even if the pair doesn't exist.

```
430 emit FeeParametersSet(
```

CVF-100. FIXED

- **Category** Documentation
- **Source** LBFactory.sol

Description This comment is confusing as there is no division by 1e14 around.

Recommendation Consider either rephrasing the code or actually performing the division.

```
495 // The result should use 4 decimals, so we divide it by 1e14
```

CVF-102. FIXED

- **Category** Suboptimal
- **Source** LBFactoryHelper.sol

Description Deploying a separate instance of the LBPair contract for every pair consumes lots of gas.

Recommendation Consider deplying a minimal proxy instead, as described in EIP-1167.

```
41 return new LBPair{salt: _salt}(factory, _tokenX, _tokenY, _activeId,
   ↪  _sampleLifetime, _packedFeeParameters);
```

CVF-122. FIXED

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

Description Underflow is possible here.

Recommendation Consider explicitly requiring the current timestamp to be not below the last update timestamp.

```
45 uint256 _deltaTime = _currentTimestamp - timestamp(_lastSample);
```

CVF-135. FIXED

- **Category** Suboptimal
- **Source** Math512Bits.sol

Description This requirement seems too strict, as x * y could be up to 512 bits long, thus shift up to 511 bits does make sense.

Recommendation Consider allowing values up to 511.

```
118 /// - The offset needs to be strictly lower than 256.
```

CVF-136. FIXED

- **Category** Procedural
- **Source** Math512Bits.sol

Description The comment says that “offset” should be strictly lower than 256, while the code allows offset that are lower or equal to 256.

Recommendation Consider either fixing the comment or making the code consistent with the comment.

```
118 /// - The offset needs to be strictly lower than 256.
```

```
136         if (offset > 256) revert Math512Bits__OffsetOverflows(offset
               ↪ );
```

CVF-137. FIXED

- **Category** Procedural
- **Source** Math512Bits.sol

Description This argument makes the function more complicated and less efficient, especially for the wound down case.

Recommendation Consider implementing two separate functions, one that rounds down and another that rounds up.

```
133 bool roundDown
```

```
190 bool roundDown
```

CVF-139. FIXED

- **Category** Procedural
- **Source** Math512Bits.sol

Description The comment says that “offset” should be strictly lower than 256, while the code allows offset that are lower or equal to 256.

Recommendation Consider either fixing the comment or making the code consistent with the comment.

```
175 /// - The offset needs to be strictly lower than 256.
```

```
193         if (offset > 256) revert Math512Bits__OffsetOverflows(offset
               ↪ );
```

CVF-161. FIXED

- **Category** Flaw
- **Source** BitMath.sol

Description There are no range checks for the “_bit” arguments.

Recommendation Consider adding appropriate checks.

```
13     uint256 _bit,
```

```
41 function closestBitRight(uint256 x, uint256 bit) internal pure
      ↪ returns (uint256 id) {
```

```
57 function closestBitLeft(uint256 x, uint256 bit) internal pure
      ↪ returns (uint256 id) {
```

CVF-164. FIXED

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

Description Overflow is possible here.

```
20 return closestBitLeft(_integer, _bit + 1);
```

CVF-176. FIXED

- **Category** Flaw
- **Source** BinHelper.sol

Description There is no range check for the arguments.

Recommendation Consider adding appropriate checks.

```
21 function getIdFromPrice(uint256 _price, uint256 _bp) internal pure
   ↪ returns (uint24) {
```

CVF-184. INFO

- **Category** Suboptimal
- **Source** TreeMath.sol

Description This argument makes the function more complicated and less efficient.

Recommendation Consider implementing two separate functions: one to search left and another to search right.

Client Comment *This can't be done cause of contract size*

```
22 bool _rightSide
```

CVF-196. FIXED

- **Category** Documentation
- **Source** Math128×128.sol

Recommendation This comment is not true. The actual format of the returned value is signed 128.128 binary fixed point.

```
27 /// @return result The binary logarithm as a signed 40.36-decimal
   ↪ fixed-point number.
```

CVF-197. FIXED

- **Category** Flaw
- **Source** Math128×128.sol

Description There is no range check for the “x” argument.

Recommendation Consider adding an appropriate check.

```
28 function log2(uint256 x) internal pure returns (int256 result) {
```

CVF-198. INFO

- **Category** Suboptimal
- **Source** Math128×128.sol

Description This function calculates logarithms with 128-bits precision, which seems too high for virtually any usage.

Recommendation Consider calculating with lower precision to save gas.

Client Comment *Louis: The function needs more than 96 bits in order to be accurate enough*

```
28 function log2(uint256 x) internal pure returns (int256 result) {
```

CVF-200. INFO

- **Category** Suboptimal
- **Source** Math128×128.sol

Recommendation Unrolling this loop would significantly reduce the gas cost.

Client Comment *Louis: The contract would be too heavy if we did that*

```
57 for (int256 delta = int256(1 << (Constants.SCALE_OFFSET - 1)); delta
   ↪  > 0; delta >>= 1) {
```

CVF-214.FIXED

- **Category** Flaw
- **Source** ILBToken.sol

Description This interface doesn’t define the “safeTransferFrom” and “balanceOfBatch” functions defined by ERC-1155.

Recommendation Consider defining these functions for compatibility with ERC-1155.

```
5 interface ILBToken {
```

8 Moderate Issues

CVF-12. INFO

- **Category** Flaw
- **Source** LBRouter.sol

Description Despite the comments, these functions don't guarantee exact outputs, but only guarantee that the actual output will not be less than the desired output.

Recommendation Consider either explaining this in the comment or fixing the code to guarantee exact output.

Client Comment *This terminology comes from uniswap*

394
```
/// @notice Swaps tokens for exact tokens while performing safety
   ↪ checks
```

421
```
/// @notice Swaps tokens for exact AVAX while performing safety
   ↪ checks
```

CVF-16. FIXED

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

Recommendation Should be "||" instead of "&&".

623
```
if (_liq.deltaIds.length != _liq.distributionX.length && _liq.
   ↪ deltaIds.length != _liq.distributionY.length)
```

626
```
if (_liq.activeIdDesired > type(uint24).max && _liq.idSlippage >
   ↪ type(uint24).max)
```

CVF-40. FIXED

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

Description Checked math is commonly used as a second layer of defence, just in case there is a bug in the code that makes overflow/underflow possible. Using unchecked math makes code much more prone to overflow-based attacks.

Recommendation Consider using unchecked math only when performance increase is significant and making unchecked blocks as small as possible.

Client Comment *Reduced them in size, and removed some*

```
241 unchecked {
305 unchecked {
387 unchecked {
479 unchecked {
590 unchecked {
657 unchecked {
693 unchecked {
745 unchecked {
835 unchecked {
855 unchecked {
```

CVF-50. FIXED

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

Description Overflow is possible when performing shift.

Recommendation Consider explicitly checking for overflow or explaining why overflow will never happen here.

```
457 (uint256(_feesX.total - _feesX.protocol) << Constants.SCALE_OFFSET)
      ↪ /
```

```
460 (uint256(_feesY.total - _feesY.protocol) << Constants.SCALE_OFFSET)
      ↪ /
```

CVF-54. FIXED

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

Description Overflow is possible here.

Recommendation Consider using checked math or explaining why overflow will never happen here.

```
516 _mintInfo.amount = (_mintInfo.amountXIn * _distribution) / Constants
      ↪ .PRECISION;
```

```
532 _mintInfo.amount = (_mintInfo.amountYIn * _distribution) / Constants
      ↪ .PRECISION;
```

CVF-63. FIXED

- **Category** Flaw
- **Source** LBPair.sol

Description There are no validity checks for the argument.

Recommendation Consider adding appropriate checks.

Client Comment *This is validated by the factory itself*

```
728 function setFeesParameters(bytes32 _packedFeeParameters) external
      ↪ override OnlyFactory {
```

CVF-67. FIXED

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

Description Overflow is possible here.

Recommendation Consider explicitly checking that "_newSize" does fit into 16 bits.

```
859 _pairInformation.oracleSize = uint16(_newSize);
```

CVF-90. FIXED

- **Category** Flaw
- **Source** LBFactory.sol

Description There is no range check for this argument.

Recommendation Consider adding an appropriate check.

```
229 uint16 _binStep
```

```
317 uint8 _binStep,
```

CVF-95. FIXED

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

Description Owner may deny user to withdraw their liquidity. Is this okay?

```
291 function setLBPairBlacklist(
```

CVF-107. INFO

- **Category** Flaw
- **Source** LBToken.sol

Description This function doesn't call the "onERC1155BatchReceived" function on a recipient smart contract.

Recommendation Consider calling it for compatibility with ERC-1155.

Client Comment *This is wanted, no callback for safety reasons*

```
117 function safeBatchTransferFrom(
```

CVF-110. FIXED

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

Description In case "_amount" is zero, this allows adding an ID with zero balance to "_userIds".

```
151     _userIds[_to].add(_id);
```

```
178 _userIds[_account].add(_id);
```

CVF-111. INFO

- **Category** Flaw
- **Source** LBToken.sol

Description This function doesn't call the " onERC1155Received" function on a recipient smart contract.

Recommendation Consider calling it for compatibility with ERC-1155.

Client Comment *This is wanted, no callback for safety reasons*

```
162 function _mint(
```

CVF-120. FIXED

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

Description Overflow is possible in the shift operation.

Recommendation Consider performing shift in 256 bits, rather than in 128 bits.

```
92 uint256 tokenPerShare = (uint256(fees.total - fees.protocol) <<
   ↪ Constants.SCALE_OFFSET) / totalSupply;
```

CVF-146. FIXED

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

Description Overflow is possible here.

Recommendation Consider calculating in 256-bit numbers.

```
53     _accumulator = (_fp.accumulator * _fp.reductionFactor) /
       ↪ Constants.HUNDRED_PERCENT;
```

```
97 return (_fp.reductionFactor * ((_acc * _fp.binStep) * (_acc * _fp.
   ↪ binStep)));
```

CVF-147. FIXED

- **Category** Flaw
- **Source** FeeHelper.sol

Description The result may exceed "_fp.maxAccumulator".

Recommendation Consider adding an explicit "require" statement to prevent this.

```
53 _accumulator = (_fp.accumulator * _fp.reductionFactor) / Constants.
      ↪ HUNDRED_PERCENT;
```

CVF-148. FIXED

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

Description Overflow is possible here.

Recommendation Consider using safe cast.

```
56 _fp.accumulator = uint72(_accumulator);
```

CVF-150. FIXED

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

Description Overflow is possible in multiplication.

Recommendation Consider using safe math or explicitly checking that "_binCrossed" is not too large.

```
66 uint256 _accumulator = uint256(_fp.accumulator) + _binCrossed *
      ↪ Constants.BASIS_POINT_MAX;
```

```
92 uint256 _acc = _fp.accumulator + _binCrossed * Constants.
      ↪ BASIS_POINT_MAX;
```

CVF-155. INFO

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

Description Overflow is possible here.

Recommendation Consider using checked math.

```
113 _feeBP = getBaseFee(_fp) + getVariableFee(_fp, _binCrossed);
```

```
130 _feeBP = getBaseFee(_fp) + getVariableFee(_fp, _binCrossed);
```

CVF-157. INFO

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

Description Overflow is possible here.

Recommendation Consider using safe cast.

```
146 fees.protocol = uint128((_fees * _fp.protocolShare) / Constants.
    ↪ HUNDRED_PERCENT);
```

CVF-162. FIXED

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

Description Underflow is possible here.

```
18 return closestBitRight(_integer, _bit - 1);
```

CVF-167. INFO

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

Description Underflow is possible here.

Client Comment *x has been shifted by inverseBit (255 - bit), all bits inferior to inverseBit are 0s and mostSignificantBit(x) can't be smaller than inverseBit*

```
43 x <<= 255 - bit;
```

```
47 return mostSignificantBit(x) - (255 - bit);
```

CVF-175. INFO

- **Category** Flaw
- **Source** BinHelper.sol

Description Due to rounding errors it is not guaranteed that the price returned by the "getPriceFromId" actually corresponds to this ID according to the "getIdFromPrice" function.

Client Comment *This is by design, but the price will always be close to the right price. Exactly at +-binStep/100% from the price*

```
21 function getIdFromPrice(uint256 _price, uint256 _bp) internal pure
    ↪ returns (uint24) {
```

```
37 function getPriceFromId(uint256 _id, uint256 _bp) internal pure
    ↪ returns (uint256) {
```

CVF-182. FIXED

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

Description Overflow is possible here.

Recommendation Consider using checked math.

```
50 return Constants.SCALE + (_bp << Constants.SCALE_OFFSET) / Constants
      ↪ .BASIS_POINT_MAX;
```

CVF-203. FIXED

- **Category** Overflow/Underflow
- **Source** Math128×128.sol

Description Overflow is possible here.

Recommendation Consider using checked multiplication.

```
86  if (absY & 0x1 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
    if (absY & 0x2 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
90  if (absY & 0x4 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
    if (absY & 0x8 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
    if (absY & 0x10 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
    if (absY & 0x20 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
    if (absY & 0x40 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
100 if (absY & 0x80 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
    if (absY & 0x100 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
    if (absY & 0x200 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
    if (absY & 0x400 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
    if (absY & 0x800 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
110 if (absY & 0x1000 != 0) result = (result * pow) >> Constants.
       ↪ SCALE_OFFSET;
    pow = (pow * pow) >> Constants.SCALE_OFFSET;
```

CVF-210. FIXED

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

Description Overflow and underflow are possible here.

Recommendation Consider using checked math.

```
32 result = (x + n - y) % n;
```

9 Minor Issues

CVF-1. FIXED

- **Category** Procedural
- **Source** LBRouter.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.

Recommendation Consider specifying as "^0.8.0". Also relevant for the next files: LBPair.sol, LBFactory.sol, LBFactoryHelper.sol, LBToken.sol, TokenHelper.sol, SwapHelper.sol, Samples.sol, PendingOwnable.sol, Math512Bits.sol, FeeHelper.sol, Constants.sol, BitMath.sol, BinHelper.sol, TreeMath.sol, SafeMath.sol, ReentrancyGuard.sol, Math128×128.sol, Encoder.sol, Decoder.sol, Buffer.sol, SafeCast.sol.

```
3 pragma solidity 0.8.9;
```

CVF-2. FIXED

- **Category** Suboptimal
- **Source** LBRouter.sol

Description Declaring top-level errors in a file named after a contract makes it h harder to navigate through the code.

Recommendation Consider either moving the error declarations into the contract or moving them into a separate file named “errors.sol”.

```
17 error LBRouter__SenderIsNotWAVAX();
   error LBRouter__PairNotCreated(IERC20 tokenX, IERC20 tokenY, uint256
      ↪  binStep);
   error LBRouter__WrongAmounts(uint256 amount, uint256 reserve);
20 error LBRouter__SwapOverflows(uint256 id);
   error LBRouter__BrokenSwapSafetyCheck();
   error LBRouter__NotFactoryOwner();
   error LBRouter__TooMuchTokensIn(uint256 excess);
   error LBRouter__BinReserveOverflows(uint256 id);
   error LBRouter__IdOverflows(int256 id);
   error LBRouter__LengthsMismatch();
   error LBRouter__IdSlippageCaught(uint256 activeIdDesired, uint256
      ↪ idSlippage, uint256 activeId);
   error LBRouter__AmountSlippageCaught(uint256 amountXMin, uint256
      ↪ amountX, uint256 amountYMin, uint256 amountY);
   error LBRouter__IdDesiredOverflows(uint256 idDesired, uint256
      ↪ idSlippage);
30 error LBRouter__FailedToSendAVAX(address recipient, uint256 amount);
   error LBRouter__DeadlineExceeded(uint256 deadline, uint256
      ↪ currentTimestamp);
   error LBRouter__AmountSlippageBPTooBig(uint256 amountSlippage);
   error LBRouter__InsufficientAmountOut(uint256 amountOutMin, uint256
      ↪ amountOut);
   error LBRouter__MaxAmountInExceeded(uint256 amountInMax, uint256
      ↪ amountIn);
   error LBRouter__InvalidTokenPath(IERC20 wrongToken);
   error LBRouter__InvalidVersion(uint256 version);
   error LBRouter__LBPairBlacklisted(ILBPair LBPair);
```

CVF-3. FIXED

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

Recommendation TokenHelper should be used here.

```
40 using SafeERC20 for IERC20;
```

CVF-5. INFO

- **Category** Suboptimal
- **Source** LBRouter.sol

Description This condition may never be true, as the loop above is only terminated when "_amountOut" is zero.

Client Comment *It's a safety check*

```
154 if (_amountOut != 0) revert LBRouter__BrokenSwapSafetyCheck(); //
    ↪ Safety check, but should never be false as it would have
    ↪ reverted on transfer
```

CVF-6. INFO

- **Category** Suboptimal
- **Source** LBRouter.sol

Description This condition may never be true, as the loop above is only terminated when "_amountIn" is zero.

Client Comment *It's a safety check*

```
203 if (_amountIn != 0) revert LBRouter__TooMuchTokensIn(_amountIn);
```

CVF-7. INFO

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

Description Zero "tokenY" address is a special value for the "_addLiquidity" function.

Recommendation Consider adding an explicit check that "_liquieidtyParameters.tokenY" is not zero.

Client Comment *This is no longer necessary, we removed the zero address thing*

```
224 _addLiquidity(_liquidityParameters);
```

CVF-8. FIXED

- **Category** Procedural
- **Source** LBRouter.sol

Recommendation Resolve TODOs in the code.

```
231 //TODO swap tokens if avax isn't Y
```

```
234 _liquidityParameters.amountYMin = msg.value; //TODO assert that
```

CVF-9. FIXED

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

Recommendation Require this attribute to be already set to the correct value to mitigate misusing.

```
232 _liquidityParameters.tokenY = IERC20(address(0));
```

CVF-10. FIXED

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

Description These names are confusing, as "_amountAVAXMin" may contain the minimum amount of non-AVAX token, and "_amountTokenMin" may contain the minimum amount for the AVAX token.

Recommendation Consider using "_amountXMin" and "_amountYMin" instead.

```
302 _amountTokenMin,
    _amountAVAXMin,
```

CVF-11. FIXED

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

Description This function should return actual amountIn and amountOut.

```
332 ) external override ensure(_deadline) verifyInputs(_pairBinSteps,
      ↪ _tokenPath) {
```

```
356 ) external override ensure(_deadline) verifyInputs(_pairBinSteps,
      ↪ _tokenPath) {
```

```
384 ) external payable override ensure(_deadline) verifyInputs(
      ↪ _pairBinSteps, _tokenPath) {
```

```
401 function swapTokensForExactTokens(
```

```
435 ) external override ensure(_deadline) verifyInputs(_pairBinSteps,
      ↪ _tokenPath) {
```

```
473 ) external payable override ensure(_deadline) verifyInputs(
      ↪ _pairBinSteps, _tokenPath) {
```

CVF-13. FIXED

- **Category** Suboptimal
- **Source** LBRouter.sol

Recommendation You can create the same functions to sweep ERC721 and ERC1155 tokens.

```
586 function sweep(
```

CVF-14. FIXED

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

Description Why do you reset it here but not in the external methods?

Recommendation Consider setting right values from the start.

```
604 if (_liq.tokenY == IERC20(address(0))) {
        _liq.tokenY = IERC20(wavax);
        _liq.amountY = msg.value;
    }
```

CVF-15. FIXED

- **Category** Procedural
- **Source** LBRouter.sol

Recommendation This should be done in the “addLiquidityAVAX” function.

```
613 _wavaxDepositAndTransfer(_liq.amountY, address(_LBPair));
```

CVF-17. INFO

- **Category** Suboptimal
- **Source** LBRouter.sol

Description This is an expensive call but you load specific storage.

Recommendation Consider optimization.

```
629 (, , uint256 _activeId) = _LBPair.getReservesAndId();
```

CVF-18. INFO

- **Category** Flaw
- **Source** LBRouter.sol

Description There are no length checks for these arguments.

Recommendation Consider adding appropriate length checks.

Client Comment *The length are checked by the function that calls the private functions*

660
```
uint256[] memory _pairBinSteps,
address[] memory _pairs,
IERC20[] memory _tokenPath,
```

722
```
address[] memory _pairs,
uint256[] memory _pairBinSteps,
IERC20[] memory _tokenPath,
```

775
```
address[] memory _pairs,
uint256[] memory _pairBinSteps,
IERC20[] memory _tokenPath,
uint256[] memory _amountsIn,
```

823
```
address[] memory _pairs,
uint256[] memory _pairBinSteps,
IERC20[] memory _tokenPath,
```

CVF-19. INFO

- **Category** Suboptimal
- **Source** LBRouter.sol

Description This line divides rounding down and then adds 1 to the result.

Recommendation Consider doing proper rounding up instead.

Client Comment *This is the way uniswap V2 is doing things and since we're estimating the outcome of a swap in an uniV2 pair, we should keep it that way*

683
```
amountsIn[i - 1] = (_reserveIn * amountOut_ * 1_000) / (_reserveOut
   ↪ - amountOut_ * 997) + 1;
```

CVF-20. INFO

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

Recommendation The values “997” and “1000” should be named constants.

Client Comment *We prefered to leave it exactly like uniswap V2 contracts, people are used to read the formulas like this*

```
683 amountsIn[i - 1] = (_reserveIn * amountOut_ * 1_000) / (_reserveOut
       ↪ - amountOut_ * 997) + 1;
```

```
749         amountOut = (_reserve1 * amountOut * 997) / (_reserve0 * 1
               ↪ _000 + amountOut * 997);
```

```
752         amountOut = (_reserve0 * amountOut * 997) / (_reserve1 * 1
               ↪ _000 + amountOut * 997);
```

```
849         uint256 _amountOut = (_reserve1 * (_balance - _reserve0) *
               ↪ 997) / (_balance * 1_000);
```

```
854         uint256 _amountOut = (_reserve0 * (_balance - _reserve1) *
               ↪ 997) / (_balance * 1_000);
```

CVF-21. INFO

- **Category** Suboptimal
- **Source** LBRouter.sol

Recommendation The type cast to “ILBToken” wouldn’t be necessary if the “ILBPair” interface would inherit the “ILBToke” interface.

```
707 ILBToken(address(_LBPair)).safeBatchTransferFrom(msg.sender, address
       ↪ (_LBPair), _ids, _amounts);
```

CVF-22. INFO

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

Description Overflow is possible here

Client Comment *We are keeping the same calculations as uniswap V2*

```
749 amountOut = (_reserve1 * amountOut * 997) / (_reserve0 * 1_000 +
       ↪ amountOut * 997);
```

CVF-23. FIXED

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

Description Despite the name, this function returns the pair itself rather then info about it.

```
871 function _getLBPairInfo(
```

CVF-25. INFO

- **Category** Flaw
- **Source** LBRouter.sol

Description There is no length check for the arguments.

Recommendation Consider adding an appropriate length check.

Client Comment *We added verifyInputs on every functions that uses this*

```
899 function _getPairs(uint256[] memory _pairBinSteps, IERC20[] memory
    ↪ _tokenPath)
```

CVF-26. FIXED

- **Category** Readability
- **Source** LBPair.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through the code.

Recommendation Consider either moving the error declarations into the contract or moving them into a separate file named “errors.sol”.

```
23 error LBPair__InsufficientAmounts();
   error LBPair__BrokenSwapSafetyCheck();
   error LBPair__BrokenMintSafetyCheck(uint256 totalDistributionX,
      ↪ uint256 totalDistributionY);
   error LBPair__InsufficientLiquidityMinted(uint256 id);
   error LBPair__InsufficientLiquidityBurned(uint256 id);
   error LBPair__WrongLengths();
   error LBPair__FlashLoanUnderflow(uint256 expectedBalance, uint256
      ↪ balance);
30 error LBPair__OnlyStrictlyIncreasingId();
   error LBPair__OnlyFactory();
   error LBPair__DistributionOverflow(uint256 id, uint256 distribution)
      ↪ ;
   error LBPair__OnlyFeeRecipient(address feeRecipient, address sender)
      ↪ ;
   error LBPair__OracleNotEnoughSample();
```

CVF-27. FIXED

- **Category** Documentation
- **Source** LBPair.sol

Recommendation Typo in the word Implementation.

```
38 /// @notice Implemention of pair
```

CVF-28. INFO

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

Recommendation The value “65536’ should be a named constant.

Client Comment *We chose to keep it like this because we can’t share the constant between LBPair.sol and Oracle.sol and we prefer not to declare twice the same variable*

```
49 using Oracle for bytes32[65_536];
```

```
111 bytes32[65_536] private _oracle;
```

CVF-29. FIXED

- **Category** Documentation
- **Source** LBPair.sol

Description It is unclear whether the “amountX” and “amountY” parameters are input our output amounts. Also it is unclear how to determinte the swap direction.

Recommendation Consider explaining in a documentation comment and/or giving the parameters more specific names.

```
53 event Swap(address indexed sender, address indexed recipient, uint24
   ↪  indexed _id, uint256 amountX, uint256 amountY);
```

CVF-30. INFO

- **Category** Suboptimal
- **Source** LBPair.sol

Description Only one of the “amountX” and “amountY” parameters is actually used each time.

Recommendation Consider leaving only one argument and adding a flag to specify the swap direction.

Client Comment *We find it easier like that*

```
53 event Swap(address indexed sender, address indexed recipient, uint24
   ↪  indexed _id, uint256 amountX, uint256 amountY);
```

CVF-31. FIXED

- **Category** Procedural
- **Source** LBPair.sol

Recommendation These events should be declared in the “ILBPair” interface.

```
53 event Swap(address indexed sender, address indexed recipient, uint24
   ↪  indexed _id, uint256 amountX, uint256 amountY);

55 event FlashLoan(

64 event Mint(

75 event Burn(address indexed sender, address indexed recipient,
   ↪ uint256[] ids, uint256[] amounts);

77 event FeesCollected(address indexed sender, address indexed
   ↪ recipient, uint256 amountX, uint256 amountY);

79 event ProtocolFeeCollected(address indexed sender, address indexed
   ↪ recipient, uint256 amountX, uint256 amountY);

81 event OracleSizeIncreased(uint256 previousSize, uint256 newSize);
```

CVF-32. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Recommendation This event should also include the liquidity amounts minted for each bin.

```
64 event Mint(
```

CVF-33. INFO

- **Category** Suboptimal
- **Source** LBPair.sol

Recommendation It would be more efficient to have a single array of structs with three fields, rather than three parallel arrays.

```
68 uint256[] ids,
   uint256[] distributionX,
70 uint256[] distributionY,
```

CVF-34. INFO

- **Category** Suboptimal
- **Source** LBPair.sol

Recommendation The “previousSize” parameter is redundant, as its value could be derived from the previous events.

```
81 event OracleSizeIncreased(uint256 previousSize, uint256 newSize);
```

CVF-36. INFO

- **Category** Flaw
- **Source** LBPair.sol

Description There is no range check for this argument.

Recommendation Consider adding an appropriate check.

```
141 uint16 _sampleLifetime,
```

CVF-38. INFO

- **Category** Readability
- **Source** LBPair.sol

Description This function implements functionality that could be implemented in plain Solidity using structs.

Recommendation Consider using plain Solidity for simplicity and readability.

```
161 function getReservesAndId()
```

CVF-39. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Description Using of _ago is unclear.

Recommendation Consider passing _lookUpTimestamp instead.

```
229 function getOracleSampleFrom(uint256 _ago)
```

CVF-41. FIXED

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

Description Overflow is possible here.

Recommendation Consider either using checked math, or explaining why overflow here will never happed or is actually fine.

```
257 cumulativeId += _pairInformation.activeId * _delta;
    cumulativeAccumulator += _fp.accumulator * _delta;
```

CVF-42. FIXED

- **Category** Readability
- **Source** LBPair.sol

Description The logic of this assembly block could be implemented in plain Solidity.

Recommendation Consider using plain Solidity for simplicity and readability.

```
284 assembly {
```

CVF-43. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Recommendation Binary “AND” here is redundant, as the first slot of the “Bin” structure doesn’t contain anything other than two reserve amounts.

```
290 reserveY := and(shr(112, _data), _mask)
```

CVF-44. INFO

- **Category** Suboptimal
- **Source** LBPair.sol

Description It’s not clear in which case this condition could be triggered.

Recommendation Consider removal.

Client Comment *Assert the uniqueness*

```
313 if (_lastId >= _id && i != 0) revert
       ↪ LBPair__OnlyStrictlyIncreasingId();
```

CVF-45. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Recommendation Consider optimizations because you do double SLOAD of the same values.

```
361 (uint256 _amountInToBin, uint256 _amountOutOfBin, FeeHelper.
    ↪ FeesDistribution memory _fees) = _pair
     .getAmounts(_bin, _fp, _swapForY, _startId, _amountIn);

_pair.updateLiquidity(
    _bin,
    _fees,
    _swapForY,
    totalSupply(_pair.activeId),
    _amountInToBin.safe112(),
370 _amountOutOfBin
);
```

CVF-46. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Description This code relies of the fact, that a value returned be the “findFirstBin” function always fits into 24 bits. Such implicit relationships make the code fragile.

Recommendation Consider using safe conversion or even better, change the return type of the “findFirstBin” function to “uint24’.

```
380 _pair.activeId = uint24(_tree.findFirstBin(_pair.activeId, _swapForY
    ↪ ));
```

CVF-47. FIXED

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

Recommendation This event should include the input amount, not the output amount.

```
423 emit Swap(msg.sender, _to, _pair.activeId, amountXOut, amountYOut);
```

CVF-51. FIXED

- **Category** Procedural
- **Source** LBPair.sol

Recommendation Usually, 100% share is represented as 1.0, rather than 100.0.

```
468 /// @param _distributionX The distribution of tokenX with sum(
       ↪ _distributionX) = 100e18 (100%) or 0 (0%)
    /// @param _distributionY The distribution of tokenY with sum(
       ↪ _distributionY) = 100e18 (100%) or 0 (0%)
```

CVF-52. FIXED

- **Category** Documentation
- **Source** LBPair.sol

Description The comment says a sum of distributions should be either 100% or 0%, while the code allows any value not exceeding 100%.

Recommendation Consider making the comment and the code consistent with each other.

```
468 /// @param _distributionX The distribution of tokenX with sum(
       ↪ _distributionX) = 100e18 (100%) or 0 (0%)
    /// @param _distributionY The distribution of tokenY with sum(
       ↪ _distributionY) = 100e18 (100%) or 0 (0%)
```

```
550             _mintInfo.totalDistributionX > 100 * Constants.PRECISION
                   ↪  ||
                _mintInfo.totalDistributionY > 100 * Constants.PRECISION
```

CVF-53. FIXED

- **Category** Procedural
- **Source** LBPair.sol

Recommendation This part is not clear and could be moved to the TreeMath file.

```
497 if (_bin.reserveX == 0 && _bin.reserveY == 0) {
        // add 1 at the right indices if the _pairInformation was empty
        uint256 _idDepth2 = _mintInfo.id / 256;
500     uint256 _idDepth1 = _mintInfo.id / 65_536;

        _tree[2][_idDepth2] |= 1 << (_mintInfo.id % 256);
        _tree[1][_idDepth1] |= 1 << (_idDepth2 % 256);
        _tree[0][0] |= 1 << _idDepth1;
    }
```

CVF-55. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Description Only 1 storage slot of the struct is changed.

Recommendation Consider optimisations of SSTORE

```
545 _bins[_mintInfo.id] = _bin;
```

CVF-56. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Recommendation Consider adding a lengths check.

```
586 uint256[] memory _ids,
    uint256[] memory _amounts,
```

CVF-57. FIXED

- **Category** Procedural
- **Source** LBPair.sol

Description This code is not clear.

Recommendation Consider moving this functionality to the TreeMath file.

```
619 if (_bin.reserveX == 0 && _bin.reserveY == 0) {
620     // removes 1 at the right indices
        uint256 _idDepth2 = _id / 256;
        uint256 _newLeafValue = _tree[2][_idDepth2] & (type(uint256).max
            ↪  - (1 << (_id % 256)));
        _tree[2][_idDepth2] = _newLeafValue;
        if (_newLeafValue == 0) {
            uint256 _idDepth1 = _id / 65_536;
            _newLeafValue = _tree[1][_idDepth1] & (type(uint256).max -
                ↪ (1 << (_idDepth2 % 256)));
            _tree[1][_idDepth1] = _newLeafValue;
            if (_newLeafValue == 0) {
                _tree[0][0] &= type(uint256).max - (1 << _idDepth1);
630         }
```

CVF-58. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Recommendation Right shift would be more efficient.

```
621 uint256 _idDepth2 = _id / 256;
```

```
625     uint256 _idDepth1 = _id / 65_536;
```

CVF-59. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Recommendation It would be more efficient to use bitwise “NOT” instead of subtraction and bitwise “AND” instead of reminder.

```
622 uint256 _newLeafValue = _tree[2][_idDepth2] & (type(uint256).max -
    ↪ (1 << (_id % 256)));
```

```
626     _newLeafValue = _tree[1][_idDepth1] & (type(uint256).max - (1 <<
        ↪  (_idDepth2 % 256)));
```

```
629         _tree[0][0] &= type(uint256).max - (1 << _idDepth1);
```

CVF-60. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Description Only 1 storage slot of the struct is changed.

Recommendation Consider optimisations of SSTORE.

```
634 _bins[_id] = _bin;
```

CVF-61. INFO

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

Recommendation This function is open for everyone but operates over the general Pair parameters. It should be explained why.

Client Comment *The size will be increased by people that wants the Oracle to be increased. They will be the one paying gas for that.*

```
649 function increaseOracleLength(uint16 _nb) external override {
```

CVF-62. INFO

- **Category** Suboptimal
- **Source** LBPair.sol

Description These two calls internally calculate the current cumulative fees for the user twice.

Recommendation Consider refactoring to avoid calculating the same value twice.

```
670 _collectFees(_fees, _bin, _account, _id, _balance);
    _updateUserDebts(_bin, _account, _id, _balance);
```

CVF-64. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Description The “from” fees are updated in case “from” and “to” are the same addresses.

Recommendation Consider not updating in such a case.

```
750 if (_from != address(0) && _from != address(this)) {
```

CVF-65. FIXED

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

Recommendation These variables could be turned into constants.

```
843 uint256 _mask112 = type(uint112).max;
    uint256 _mask144 = type(uint144).max;
```

CVF-66. FIXED

- **Category** Suboptimal
- **Source** LBPair.sol

Description Specifying how many samples to add to the oracle size, rather than the desired new oracle size, could lead to oversized oracles. If several users will simultaneously ry to increate the oracle size, then all the increases will be applied, while actually only the biggest increase would be needed to apply.

Recommendation Consider accepting the new size as an argument.

```
853 /// @param _nb The number of sample to add to the oracle
```

CVF-68. FIXED

- **Category** Procedural
- **Source** LBFactory.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navogate through the code.

Recommendation Consider either moving the error declarations into the contract or moving them into a separate file named “errors.sol”.

```
error LBFactory__IdenticalAddresses(IERC20 token);
error LBFactory__ZeroAddress();
error LBFactory__FactoryHelperAlreadyInitialized();
error LBFactory__LBPairAlreadyExists(IERC20 tokenX, IERC20 tokenY,
    ↪ uint256 _binStep);
error LBFactory__DecreasingPeriods(uint16 filterPeriod, uint16
    ↪ decayPeriod);
error LBFactory__BaseFactorOverflows(uint16 baseFactor, uint256 max)
    ↪ ;
error LBFactory__ReductionFactorOverflows(uint16 reductionFactor,
    ↪ uint256 max);
error LBFactory__VariableFeeControlOverflows(uint16
    ↪ variableFeeControl, uint256 max);
error LBFactory__BaseFeesBelowMin(uint256 baseFees, uint256
    ↪ minBaseFees);
error LBFactory__FeesAboveMax(uint256 fees, uint256 maxFees);
error LBFactory__BinStepRequirementsBreached(uint256 lowerBound,
    ↪ uint16 binStep, uint256 higherBound);
error LBFactory__ProtocolShareOverflows(uint16 protocolShare,
    ↪ uint256 max);
error LBFactory__FunctionIsLockedForUsers(address user);
error LBFactory__FactoryLockIsAlreadyInTheSameState();
error LBFactory__LBPairBlacklistIsAlreadyInTheSameState();
error LBFactory__BinStepHasNoPreset(uint256 binStep);
```

CVF-69. FIXED

- **Category** Readability
- **Source** LBFactory.sol

Recommendation “0.0001e18” would be more readable.

```
uint256 public constant override MIN_FEE = 1e14; // 0.01%
```

CVF-70. FIXED

- **Category** Readability
- **Source** LBFactory.sol

Recommendation “0.1e18” would be more readable.

```
31 uint256 public constant override MAX_FEE = 1e17; // 10%
```

CVF-71. INFO

- **Category** Documentation
- **Source** LBFactory.sol

Description What does 247 mean? Please clarify.

```
34 uint256 public constant override MAX_BIN_STEP = 100; // 1%, can't be
   ↪  greater than 247 for indexing reasons
```

```
50 // The max binStep set is 247. We use this method instead of an
   ↪ array to keep it ordered and to reduce gas
```

```
54 // The max binStep set is 247. We use this method instead of an
   ↪ array to keep it ordered and to reduce gas
```

CVF-72. FIXED

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

Description The name is confusing as it is associated with reentrancy guard functionality.

Recommendation Consider documenting.

```
43 bool public override unlocked;
```

CVF-73. FIXED

- **Category** Documentation
- **Source** LBFactory.sol

Description The semantics of the keys in this mapping in unclear.

```
47 mapping(IERC20 => mapping(IERC20 => mapping(uint256 => LBPairInfo)))
   ↪  private _LBPairsInfo;
```

CVF-75. INFO

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

Recommendation Event parameter names usually start with lower case letters.

Client Comment *We agreed that acronyms will be in caps*

```
60 event LBPairCreated(IERC20 indexed tokenX, IERC20 indexed tokenY,
      ↪ ILBPair LBPair, uint256 pid);
```

```
66     ILBPair indexed LBPair,
```

```
79 event LBPairBlacklistedStateChanged(ILBPair LBPair, bool blacklist);
```

CVF-76. INFO

- **Category** Suboptimal
- **Source** LBFactory.sol

Recommendation The “oldRecipient” parameter is redundant as its value could be derived from the previous events.

```
62 event FeeRecipientChanged(address oldRecipient, address newRecipient
      ↪ );
```

CVF-77. FIXED

- **Category** Procedural
- **Source** LBFactory.sol

Recommendation These events should be declared in the “ILBFactory” interface.

```
60 event LBPairCreated(IERC20 indexed tokenX, IERC20 indexed tokenY,
      ↪ ILBPair LBPair, uint256 pid);
```

```
62 event FeeRecipientChanged(address oldRecipient, address newRecipient
      ↪ );
```

```
64 event FeeParametersSet(
```

```
77 event FactoryLocked(bool unlocked);
```

```
79 event LBPairBlacklistedStateChanged(ILBPair LBPair, bool blacklist);
```

```
81 event PresetSet(
```

```
92 event PresetRemoved(uint256 binStep);
```

CVF-78. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Recommendation These parameters should be indexed.

```
65 address sender,
   ILBPair indexed LBPair,
```

CVF-79. INFO

- **Category** Suboptimal
- **Source** LBFactory.sol

Description The types of these event parameters differ from the values of corresponding function arguments.

Recommendation Consider using the same types for the same things for consistency.

Client Comment *Easier to use if returned as uint256*

```
67 uint256 binStep,
   uint256 baseFactor,
   uint256 filterPeriod,
70 uint256 decayPeriod,
   uint256 reductionFactor,
   uint256 variableFeeControl,
   uint256 protocolShare,
   uint256 maxAccumulator
```

```
82 uint256 indexed binStep,
   uint256 baseFactor,
   uint256 filterPeriod,
   uint256 decayPeriod,
   uint256 reductionFactor,
   uint256 variableFeeControl,
   uint256 protocolShare,
   uint256 maxAccumulator,
90 uint256 sampleLifetime
```

CVF-80. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Recommendation It would be more efficient to declare two separate events: one for lock and another for unlock.

```
77 event FactoryLocked(bool unlocked);
```

CVF-81. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Recommendation The “LBPair” parameter should be indexed.

```
79 event LBPairBlacklistedStateChanged(ILBPair LBPair, bool blacklist);
```

CVF-82. INFO

- **Category** Suboptimal
- **Source** LBFactory.sol

Recommendation It would be more efficient to declare two separate events: one for blacklisting and another for removing from the black list.

```
79 event LBPairBlacklistedStateChanged(ILBPair LBPair, bool blacklist);
```

CVF-83. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Recommendation The “binStep” parameter should be indexed.

```
92 event PresetRemoved(uint256 binStep);
```

CVF-84. FIXED

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

Recommendation This function should log an event.

Client Comment *FactoryHelper has been removed*

```
102 function setFactoryHelper() external override {
```

CVF-85. FIXED

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

Description Anyone can call this function. So frontrunners may break your contract.

Recommendation Consider protecting it.

Client Comment *FactoryHelper has been removed*

```
104 factoryHelper = ILBFactoryHelper(msg.sender);
```

CVF-86. FIXED

- **Category** Documentation
- **Source** LBFactory.sol

Recommendation This comment is incorrect, as this function actually returns a "LBPair-Info" structure rather than "LBPair" address.

```
113 /// @notice Returns the address of the LBPair if it exists,
```

CVF-87. FIXED

- **Category** Documentation
- **Source** LBFactory.sol

Description Typo in the comment.

Client Comment *Fixed somewhere*

```
127 /// @notice Vies function to return the different parameters of the
    ↪ preset
```

```
185 /// @notice Vies function to return the list of available binStep
    ↪ with a preset
```

CVF-91. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Recommendation Only one check would be needed if this line would be moved after the "_sortTokens" call.

```
235 if (address(_tokenX) == address(0) || address(_tokenY) == address(0)
    ↪ ) revert LBFactory__ZeroAddress();
```

CVF-92. INFO

- **Category** Suboptimal
- **Source** LBFactory.sol

Description The expression "_LBTokenInfo[_tokenA][_tokenB][_binStep] is calculated twice.

Recommendation Consider calculating once and reusing.

```
238 if (address(_LBPairsInfo[_tokenA][_tokenB][_binStep].LBPair) !=
    ↪ address(0))
```

```
257 _LBPairsInfo[_tokenA][_tokenB][_binStep] = LBPairInfo({
```

CVF-93. FIXED

- **Category** Readability
- **Source** LBFactory.sol

Recommendation Consider using named arguments for clarity.

```
248 _LBPair = factoryHelper.createLBPair(
        _tokenX,
250     _tokenY,
        keccak256(abi.encode(_tokenX, _tokenY, _binStep)),
        _activeId,
        uint16(_sampleLifetime),
        _preset
    );
```

CVF-94. INFO

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

Recommendation This is not clear why such salt is needed.

Client Comment *To make the address desterministic*

```
251 keccak256(abi.encode(_tokenX, _tokenY, _binStep)),
```

CVF-96. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Description The "_locked" expression is calculated twice.

Recommendation Consider calculating once and reusing.

```
387 if (unlocked == !_locked) revert
       ↪ LBFactory__FactoryLockIsAlreadyInTheSameState();
    unlocked = !_locked;
```

CVF-99. FIXED

- **Category** Suboptimal
- **Source** LBFactory.sol

Description This event is emitted even if nothing actually changed.

```
451 emit FeeRecipientChanged(oldFeeRecipient, _feeRecipient);
```

CVF-101. FIXED

- **Category** Procedural
- **Source** LBFactoryHelper.sol

Description Declaring a top-level error in a file named after a contract makes it harder to navigate through the code.

Recommendation Consider iether moving the error definition into the contract or moving it to a separate file named “errors.sol”.

```
8 error LBFactoryHelper__CallerIsNotFactory();
```

CVF-103. FIXED

- **Category** Procedural
- **Source** LBToken.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through the code.

Recommendation Consider either moving the error declarations into the contract or moving them into a separate file named “errors.sol”.

```
8  error LBToken__SpenderNotApproved(address owner, address spender);
   error LBToken__TransferFromOrToAddress0();
10 error LBToken__MintToAddress0();
   error LBToken__BurnFromAddress0();
   error LBToken__BurnExceedsBalance(address from, uint256 id, uint256
      ↪ amount);
   error LBToken__LengthMismatch(uint256 accountsLength, uint256
      ↪ idsLength);
   error LBToken__SelfApproval(address owner);
   error LBToken__TransferExceedsBalance(address from, uint256 id,
      ↪ uint256 amount);
```

CVF-104. FIXED

- **Category** Suboptimal
- **Source** LBToken.sol

Description This event is never emitted.

Recommendation Consider removing it.

```
39 event Transfer();
```

CVF-105. INFO

- **Category** Suboptimal
- **Source** LBToken.sol

Description While Solidity doesn't support immutable variables of type string, it is possible to very efficiently pack a short string into a bytes32 value and then unpack it back.

Recommendation In case names and symbols do fit into 31 bytes, consider packing them and storing in immutable variables. Here is a cose sample demonstrating how short strings could be packed/unpacked: https://gist.github.com/3sGgpQ8H/567354534170905e047b299286697e19

45
```
_name = name_;
_symbol = symbol_;
```

CVF-106. INFO

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

Recommendation This function could be defined as "pure".

Client Comment *This function have been removed*

63
```
function decimals() public view virtual override returns (uint8) {
```

CVF-108. INFO

- **Category** Suboptimal
- **Source** LBToken.sol

Recommendation This check is redundant. It is anyway possible to send to a dead address, and it is not possible to send from a zero address as zero address cannot approve anything.

Client Comment *It's OZ implementation*

126
```
if (_from == address(0) || _to == address(0)) revert
    ↪ LBToken__TransferFromOrToAddress0();
```

CVF-109. FIXED

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

Description In case "_amount' is zero, this will remove a non-existing record.

144
```
    _userIds[_from].remove(_id);
```

206
```
_userIds[_account].remove(_id);
```

CVF-112. INFO

- **Category** Suboptimal
- **Source** LBToken.sol

Description This event is emitted even if nothing actually changed.

Client Comment *This is how it is done on OpenZeppelin contracts too*

```
224 emit ApprovalForAll(_owner, _spender, _approved);
```

CVF-113. INFO

- **Category** Procedural
- **Source** LBToken.sol

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

```
257 ) internal virtual {}
```

CVF-114. FIXED

- **Category** Suboptimal
- **Source** TokenHelper.sol

Description Declaring a top-level error in a file named after a library makes it harder to navigate through the code.

Recommendation Consider either moving the error declaration into the library or moving it into a separate file named “errors.sol”.

```
7 error TokenHelper__TransferFailed(IERC20 token, address recipient,
    ↪ uint256 amount);
```

CVF-115. FIXED

- **Category** Documentation
- **Source** TokenHelper.sol

Description Wrong name.

Recommendation Consider rephrasing.

```
15 /// To use this library you can add a `using SafeERC20 for IERC20;`
    ↪ statement to your contract,
```

CVF-116. INFO

- **Category** Suboptimal
- **Source** TokenHelper.sol

Description You don't need to accumulate reserve and fees in a separate variable.

Recommendation Consider substration directly from token.balanceOf.

Client Comment *Save gas cause you don't need to check the addition*

```
50 uint256 _internalBalance;
   unchecked {
       _internalBalance = reserve + fees;
   }
   return token.balanceOf(address(this)) - _internalBalance;
```

CVF-117. FIXED

- **Category** Documentation
- **Source** SwapHelper.sol

Recommendation Should be "sent by the user".

```
23 /// @param amountIn The amount sent to the user
```

CVF-118. FIXED

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

Description In all usages of this function, startId is set to the current pair.activeId, so it's not clear why could it happend that they are not the same.

```
56 uint256 _deltaId = startId > pair.activeId ? startId - pair.activeId
   ↪  : pair.activeId - startId;
```

CVF-119. INFO

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

Recommendation This should be calculated only when amountIn > _maxAmountInToBin.

Client Comment *Seems wrong, but this has changed*

```
58 fees = fp.getFeesDistribution(fp.getFees(_maxAmountInToBin, _deltaId
   ↪ ));
```

CVF-121. FIXED

- **Category** Suboptimal
- **Source** Samples.sol

Description EVM opcode TIMESTAMP is cheaper (=2) then opcode SWAP (=3).

Recommendation Consider removing this variable.

```
44 uint256 _currentTimestamp = block.timestamp;
```

CVF-123. INFO

- **Category** Suboptimal
- **Source** Samples.sol

Description In case the last sample is not initialized, "_deltaTime" will be equal to the current timestamp.

Recommendation Consider using zero instead.

```
45 uint256 _deltaTime = _currentTimestamp - timestamp(_lastSample);
```

CVF-124. FIXED

- **Category** Suboptimal
- **Source** Samples.sol

Description The cumulative values calculated here may exceed the bit widths allocated to them in a sample.

Recommendation Consider either preventing this somehow or explaining why such overflows are fine.

```
46 uint256 _cumulativeId = cumulativeId(_lastSample) + _activeId *
   ↪ _deltaTime;
uint256 _cumulativeAccumulator = cumulativeAccumulator(_lastSample)
   ↪ + _accumulator * _deltaTime;
uint256 _cumulativeBinCrossed = cumulativeBinCrossed(_lastSample) +
   ↪ _binCrossed * _deltaTime;
```

CVF-125. FIXED

- **Category** Suboptimal
- **Source** PendingOwnable.sol

Description Declaring top-level erros in a file named after a library makes it harder to navigate through the code.

Recommendation Consider either moving the error declaraations into the library, or moving them into a separate file named “errors.sol”.

```
7  error PendingOwnable__NotOwner();
   error PendingOwnable__NotPendingOwner();
   error PendingOwnable__PendingOwnerAlreadySet();
10 error PendingOwnable__NoPendingOwner();
```

CVF-126. FIXED

- **Category** Procedural
- **Source** PendingOwnable.sol

Recommendation These events should be moved to the “IPendingOwnable” interface.

```
32 event PendingOwnerSet(address indexed pendingOwner);
   event OwnershipTransferred(address indexed previousOwner, address
      ↪ indexed newOwner);
```

CVF-127. INFO

- **Category** Suboptimal
- **Source** PendingOwnable.sol

Description The “previousOwner” parameter is redundant as its value could be derived from the previous events.

Recommendation Consider removing this parameter.

```
33 event OwnershipTransferred(address indexed previousOwner, address
      ↪ indexed newOwner);
```

CVF-128. INFO

- **Category** Flaw
- **Source** PendingOwnable.sol

Recommendation How msg.sender could be = address(0)? Consider removing.

Client Comment *It’s to be perfectionnist*

```
43 if (msg.sender != _pendingOwner || msg.sender == address(0)) revert
      ↪ PendingOwnable__NotPendingOwner();
```

CVF-129. FIXED

- **Category** Suboptimal
- **Source** PendingOwnable.sol

Description This function allows setting the pending owner to a zero address, while zero pending owner is a special value.

Recommendation Consider forbidding setting the pending owner to a zero address using this function.

```
66 function setPendingOwner(address pendingOwner_) public override
   ↪ onlyOwner {
```

CVF-130. INFO

- **Category** Suboptimal
- **Source** PendingOwnable.sol

Description This check loos redundant. The owner may anyway revoke pending owner and then set a new one.

Recommendation Consider removing this check.

```
67 if (_pendingOwner != address(0)) revert
   ↪ PendingOwnable__PendingOwnerAlreadySet();
```

CVF-131. FIXED

- **Category** Readability
- **Source** Math512Bits.sol

Description Declaring top-level errors in a file named after a library makes it harder to navigate through the code.

Recommendation Consider either moving the error declarations into the library or moving them into a separate file named “errors.sol”.

```
5 error Math512Bits__MulDivOverflow(uint256 prod1, uint256 denominator
  ↪ );
error Math512Bits__ShiftDivOverflow(uint256 prod1, uint256
  ↪ denominator);
error Math512Bits__MulShiftOverflow(uint256 prod1, uint256 offset);
error Math512Bits__OffsetOverflows(uint256 offset);
```

CVF-132. FIXED

- **Category** Suboptimal
- **Source** Math512Bits.sol

Description This “unchecked” block effectively does nothing as there is nothing to check inside.

```
44 unchecked {
```

CVF-133. INFO

- **Category** Suboptimal
- **Source** Math512Bits.sol

Recommendation This could be simplified as: uint256 lpotdod = denominator & -enominator;

Client Comment *Would require double cast*

```
74 uint256 lpotdod = denominator & (~denominator + 1);
```

```
236     uint256 lpotdod = denominator & (~denominator + 1);
```

CVF-134. FIXED

- **Category** Suboptimal
- **Source** Math512Bits.sol

Description The “+=” operator here is confusing as “result” is guaranteed to be zero here.

Recommendation Consider using “=” instead.

```
107 result += prod0 * inverse;
```

CVF-138. FIXED

- **Category** Documentation
- **Source** Math512Bits.sol

Recommendation The formula here is incorrect. The correct formula would be: (x « offset) / y

```
169 /// @notice Calculates x >> offset / y with full precision.
```

CVF-140. FIXED

- **Category** Suboptimal
- **Source** Math512Bits.sol

Description In some cases offsets up to 511 could make sense, as x « offset could still fit into 512 bits.

Recommendation Consider allowing offsets up to 511.

175
```
/// - The offset needs to be strictly lower than 256.
```

CVF-141. FIXED

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

Description It is unclear, how a non-zero value for fhis field is interpreted.

Recommendation Consider explaining.

17
```
/// - variableFeeControl: The variable fee control, used to control
   ↪ the variable fee, can be 0 to disable them
```

CVF-142. INFO

- **Category** Documentation
- **Source** FeeHelper.sol

Description The number format and the measurement units for these fields are unclear.

Recommendation Consider documenting.

23
```
uint8 binStep;
uint8 baseFactor;
uint16 filterPeriod;
uint16 decayPeriod;
uint8 reductionFactor;
uint8 variableFeeControl;
uint8 protocolShare;
```

CVF-143. FIXED

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

Description This attribute is never used, is it really required?

28
```
uint8 variableFeeControl;
```

CVF-144. FIXED

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

Description 72 is an unclear magic number here.

Recommendation Consider using of the named constant with explanations.

```
49 uint256 _accumulator; // Can't overflow as _accumulator <= _fp.
   ↪ accumulator <= _fp.maxAccumulator < 2**72
```

CVF-145. FIXED

- **Category** Suboptimal
- **Source** FeeHelper.sol

Recommendation This function is called on every swap. So if you have a lot of transactions, most of the time block.timestamp will be the same as block.timestamp. You may skip the execution in such case.

```
47 uint256 deltaT = block.timestamp - _fp.time;
```

```
51     _accumulator = _fp.accumulator;
```

CVF-149. FIXED

- **Category** Suboptimal
- **Source** FeeHelper.sol

Recommendation You don't have to declare stack variable since you can do direct set of '_fp.accumulator' in every if-branch

```
56 _fp.accumulator = uint72(_accumulator);
```

CVF-151. FIXED

- **Category** Suboptimal
- **Source** FeeHelper.sol

Recommendation Using safe cast here is redundant, as the conditional statement above guarantees that _accumulator <= _fp.maxAccumulator < 2^72.

```
70 _fp.accumulator = _accumulator.safe72();
```

CVF-152. INFO

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

Recommendation The value “1e12” should be a named constant.

```
80 return uint256(_fp.baseFactor) * _fp.binStep * 1e12;
```

CVF-153. FIXED

- **Category** Suboptimal
- **Source** FeeHelper.sol

Description The expression “_acc * _fp.binStep” is calculated twice.

Recommendation Consider calculating once and reusing.

```
97 return (_fp.reductionFactor * ((_acc * _fp.binStep) * (_acc * _fp.
   ↪ binStep)));
```

CVF-154. FIXED

- **Category** Readability
- **Source** FeeHelper.sol

Recommendation Should be ”Returns”.

```
101 /// @notice Return the fees added to an amount
```

```
118 /// @notice Return the fees from an amount
```

```
135 /// @notice Return the fees distribution added to an amount
```

CVF-156. FIXED

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

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

Recommendation Consider using the “mulDiv” function.

```
115 return (_amount * _feeBP) / (Constants.PRECISION);
```

```
132 return (_amountPlusFee * _feeBP) / (Constants.PRECISION + _feeBP);
```

CVF-158. INFO

- **Category** Suboptimal
- **Source** FeeHelper.sol

Description Overflow during multiplication here is prevented by an implicit check at the previous line.

Recommendation Consider caching the output of the “safe128” cast from the previous line and using it here, to make the relationship between these two lines explicit.

```
146 fees.protocol = uint128((_fees * _fp.protocolShare) / Constants.
      ↪ HUNDRED_PERCENT);
```

CVF-159. FIXED

- **Category** Suboptimal
- **Source** Constants.sol

Description This has a small accuracy.

Recommendation Consider using of 10_000 instead for more flexibility.

```
10 uint256 internal constant HUNDRED_PERCENT = 100;
```

CVF-160. FIXED

- **Category** Readability
- **Source** BitMath.sol

Description Here ”to the _integer” seems incorrect.

Recommendation Consider rephrasing.

```
6 /// @notice Returns the non-zero bit closest to the `_integer` to
    ↪ the right (or left) of the `bit` index
```

CVF-163. FIXED

- **Category** Readability
- **Source** BitMath.sol

Recommendation Should be “else return”.

```
20     return closestBitLeft(_integer, _bit + 1);
```

```
32 return leastSignificantBit(_integer);
```

CVF-165. FIXED

- **Category** Documentation
- **Source** BitMath.sol

Recommendation Wrong comment as the function does not return a tuple.

```
35 /// @notice Returns a tuple (uint256 id, bool found),
```

CVF-166. FIXED

- **Category** Suboptimal
- **Source** BitMath.sol

Description The expression “255 - bit” is calculated twice.

Recommendation Consider calculating once and reusing.

```
43 x <<= 255 - bit;
```

```
47 return mostSignificantBit(x) - (255 - bit);
```

CVF-168. FIXED

- **Category** Suboptimal
- **Source** BitMath.sol

Recommendation ‘msb‘ is unclear, consider using of full name like mostSignificantBitIndex.

```
69 /// @return msb The index of the most significant bit of x
70 function mostSignificantBit(uint256 x) internal pure returns (
   ↪ uint256 msb) {
```

```
109 function leastSignificantBit(uint256 x) internal pure returns (
    ↪ uint256 lsb) {
110     unchecked {
```

CVF-169. FIXED

- **Category** Suboptimal
- **Source** BitMath.sol

Description The “+=” operator is redundant here, as “msb” is guaranteed to be zero here.

Recommendation Consider using “=” instead.

```
74 msb += 128;
```

CVF-170. FIXED

- **Category** Readability
- **Source** BinHelper.sol

Description Declaring top-level erros in a file named after a library makes it harder to navigate through the code.

Recommendation Consider either moving error declarations into the library or moving them into a separate file named “errors.sol”.

```
7 error BinHelper__WrongBPValue(uint256 bp);
  error BinHelper__IdOverflows(int256 _id);
```

CVF-171. FIXED

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

Description It reads like ”shift by 24 bits” but in fact it’s powered by 23.

Recommendation Consider renaming.

```
13 int256 private constant INT24_SHIFT = 2**23;
```

CVF-172. FIXED

- **Category** Suboptimal
- **Source** BinHelper.sol

Recommendation Shifts could be intuitively written as 1«n, instead of 2**n.

```
13 int256 private constant INT24_SHIFT = 2**23;
```

CVF-173. INFO

- **Category** Documentation
- **Source** BinHelper.sol

Description This comment is confusing, what does it mean ”id may be inaccurate”.

Recommendation Consider adding more explanations.

```
16 /// @dev The id may be inaccurate due to rounding issues, always
     ↪ trust getPriceFromId rather than
```

CVF-174. INFO

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

Description 36 decimals may be not enough for tokens with a very big number of decimals.

```
18 /// @param _price The price of y per x (with 36 decimals)
```

CVF-177. INFO

- **Category** Suboptimal
- **Source** BinHelper.sol

Description As "_bp" (bit step) value is fixed for a pool, calculating 1 + _bp and even log2 (1 + _bp) every time is waste of gas.

Recommendation Consider precomputing and reusing these values.

```
23 uint256 _bpValue = _getBPValue(_bp);
```

```
25 int256 _id = INT24_SHIFT + _price.log2() / _bpValue.log2();
```

```
41 return _getBPValue(_bp).power(_realId);
```

CVF-178. FIXED

- **Category** Documentation
- **Source** BinHelper.sol

Description The comment about 36 decimals is confusing, as one could thing that this function returns result as a decimal fixed-point number with 36 decimals, while it actually returns a binary fixed-point number with 128 bits after the dot.

Recommendation Consider rephrasing.

```
32 /// @notice Returns the price corresponding to the given ID (with 36
   ↪  decimals)
```

```
36 /// @return The price corresponding to this id (with 36 decimals)
```

CVF-179. INFO

- **Category** Documentation
- **Source** BinHelper.sol

Description For each ID there is a range of prices. It is unclear what particular price from that range is returned by this function.

Recommendation Consider clarifying.

```
33 /// @dev This is the trusted function to link id to price, the other
   ↪  way may be inaccurate
```

CVF-180. FIXED

- **Category** Suboptimal
- **Source** BinHelper.sol

Recommendation Conversion to “uint256” is redundant as “_id” is already “uint256”.

```
39 int256 _realId = int256(uint256(_id)) - INT24_SHIFT;
```

CVF-181. FIXED

- **Category** Flaw
- **Source** BinHelper.sol

Description There is no range check for the argument.

Recommendation Consider adding an appropriate check.

```
48 function _getBPValue(uint256 _bp) internal pure returns (uint256) {
```

CVF-183. FIXED

- **Category** Procedural
- **Source** TreeMath.sol

Description Declaring a top-level error in a file named after a library makes it harder to navigate through the code.

Recommendation Consider either moving the error declaration into the library or moving it into a separate file named “errors.sol”.

```
7 error TreeMath__ErrorDepthSearch();
```

CVF-185. FIXED

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

Recommendation The return type for this function should be “uint24”.

```
23 ) internal view returns (uint256) {
```

CVF-186. FIXED

- **Category** Suboptimal
- **Source** TreeMath.sol

Recommendation Bitwise “AND” would be more efficient.

```
27 uint256 bit = _binId % 256;
```

```
39 bit = _binId % 256;
```

```
43 if ((_rightSide && _binId % 256 != 0) || (!_rightSide && _binId %
   ↪ 256 != 255)) {
```

CVF-187. FIXED

- **Category** Suboptimal
- **Source** TreeMath.sol

Recommendation Shift would be more efficient.

```
28 _binId /= 256;
```

```
40 _binId /= 256;
```

CVF-188. FIXED

- **Category** Suboptimal
- **Source** TreeMath.sol

Recommendation Shift would be more efficient than multiplication.

```
35          return _binId * 256 + bit;
```

```
47          _binId = 256 * _binId + bit;
```

```
50          return _binId * 256 + bit;
```

```
59  _binId = 256 * _binId + current.significantBit(_rightSide);
```

```
62  return _binId * 256 + bit;
```

CVF-189. INFO

- **Category** Procedural
- **Source** TreeMath.sol

Recommendation This assignment should be moved into the conditional operator above.

Client Comment *This is wrong, it should stay there*

```
39  bit = _binId % 256;
```

CVF-190. FIXED

- **Category** Suboptimal
- **Source** TreeMath.sol

Description The expression "_binId % 256" is calculated twice.

Recommendation Consider calculating once and reusing.

```
43  if ((_rightSide && _binId % 256 != 0) || (!_rightSide && _binId %
    ↪ 256 != 255)) {
```

CVF-191. FIXED

- **Category** Procedural
- **Source** ReentrancyGuard.sol

Description Declaring a top-level error in a file named after a library makes it harder to navigate through the code.

Recommendation Consider either moving the error declaration into the library or moving it into a separate file named “errors.sol”.

```
5 error ReentrancyGuard__ReentrantCall();
```

CVF-192. INFO

- **Category** Suboptimal
- **Source** ReentrancyGuard.sol

Recommendation This assignment is redundant, as the “nonReentrant” modifier would accept any value expect for “_ENTERED”, including a zero value.

Client Comment *OZ does this to save gas*

```
28 _status = _NOT_ENTERED;
```

CVF-193. INFO

- **Category** Suboptimal
- **Source** ReentrancyGuard.sol

Recommendation It would be more reliable to check “_statue != _NOT_ENTERED”.

```
38 if (_status == _ENTERED) revert ReentrancyGuard__ReentrantCall();
```

CVF-194. FIXED

- **Category** Procedural
- **Source** Math128×128.sol

Description Declaring top level errors in a file named after a library makes it harder to navigate through the code.

Recommendation Consider either moving the error declaration into the library or moving it into a separate file named “errors.sol”.

```
9 error Math128x128__PowerUnderflow(uint256 x, int256 y);
```

CVF-195. FIXED

- **Category** Documentation
- **Source** Math128×128.sol

Recommendation The actual format of the argument is 128.128 binary fixed point, not decimal fixed point.

```
26 /// @param x The unsigned 128.128-decimal fixed-point number for
      ↪ which to calculate the binary logarithm.
```

CVF-199. INFO

- **Category** Suboptimal
- **Source** Math128×128.sol

Description Calculating the logarithm sign and inverting e argument in case it is less than one is an unnecessary overcomplication that could lead to precision degradation.

Recommendation Just treat “x” as an integer, calculate the binary logarithm of it and then subtract 128 from the calculated logarithm value.

```
30 // This works because log2(x) = -log2(1/x).
```

CVF-201. FIXED

- **Category** Suboptimal
- **Source** Math128×128.sol

Description It “y” would be fit into 128 rather than 129 bits, it would be possible to replace this expansive “mulShift” call with plain multiplication and shift.

Recommendation Consider fitting “y” into 128 bits.

```
58 y = y.mulShift(y, Constants.SCALE_OFFSET, true);
```

CVF-202. FIXED

- **Category** Overflow/Underflow
- **Source** Math128×128.sol

Description Underflow is possible when converting “-y” to “uint256”.

```
80 uint256 absY = y >= 0 ? uint256(y) : uint256(-y);
```

CVF-204. FIXED

- **Category** Procedural
- **Source** Math128×128.sol

Recommendation The “absY > 0xffff” check should be done right after “absY” was calculated.

```
126 if (result == 0 || absY > 0xffffff) revert
       ↪ Math128x128__PowerUnderflow(x, y);
```

CVF-205. INFO

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

Description Applying mask could silently drop some bits of “value”.

Recommendation Instead, consider explicitly requiring that value & mask == mask, i.e. that no bits outside of the mask are set in the value.

```
17 sample := shl(_offset, and(_value, _mask))
```

CVF-206. INFO

- **Category** Suboptimal
- **Source** Decoder.sol

Recommendation This logic could be implemented in pure Solidity. No assembly required.

```
16 assembly {
       value := and(shr(_offset, _sample), _mask)
   }
```

CVF-207. FIXED

- **Category** Suboptimal
- **Source** Buffer.sol

Description This function is equivalent to the Solidity builtin “addmod” function.

Recommendation Consider removing this function.

```
11 function addMod(
```

CVF-208. FIXED

- **Category** Documentation
- **Source** Buffer.sol

Recommendation This function actually calculates: ((x + n - y) mod 2^256) % n

```
21 /// @notice Internal function to do positive (x - y) % n
```

CVF-211.FIXED

- **Category** Procedural
- **Source** SafeCast.sol

Description Declaring top-level errors in a file named after a library makes it harder to navigate through the code.

Recommendation Consider either moving error declarations into the library or moving them into a seaprate file named “errors.sol”.

```
error SafeCast__Exceeds256Bits(uint256 x);
error SafeCast__Exceeds248Bits(uint256 x);
error SafeCast__Exceeds240Bits(uint256 x);
error SafeCast__Exceeds232Bits(uint256 x);
error SafeCast__Exceeds224Bits(uint256 x);
error SafeCast__Exceeds216Bits(uint256 x);
error SafeCast__Exceeds208Bits(uint256 x);
error SafeCast__Exceeds200Bits(uint256 x);
error SafeCast__Exceeds192Bits(uint256 x);
error SafeCast__Exceeds184Bits(uint256 x);
error SafeCast__Exceeds176Bits(uint256 x);
error SafeCast__Exceeds168Bits(uint256 x);
error SafeCast__Exceeds160Bits(uint256 x);
error SafeCast__Exceeds152Bits(uint256 x);
error SafeCast__Exceeds144Bits(uint256 x);
error SafeCast__Exceeds136Bits(uint256 x);
error SafeCast__Exceeds128Bits(uint256 x);
error SafeCast__Exceeds120Bits(uint256 x);
error SafeCast__Exceeds112Bits(uint256 x);
error SafeCast__Exceeds104Bits(uint256 x);
error SafeCast__Exceeds96Bits(uint256 x);
error SafeCast__Exceeds88Bits(uint256 x);
error SafeCast__Exceeds80Bits(uint256 x);
error SafeCast__Exceeds72Bits(uint256 x);
error SafeCast__Exceeds64Bits(uint256 x);
error SafeCast__Exceeds56Bits(uint256 x);
error SafeCast__Exceeds48Bits(uint256 x);
error SafeCast__Exceeds40Bits(uint256 x);
error SafeCast__Exceeds32Bits(uint256 x);
error SafeCast__Exceeds24Bits(uint256 x);
error SafeCast__Exceeds16Bits(uint256 x);
error SafeCast__Exceeds8Bits(uint256 x);
```

CVF-212. FIXED

- **Category** Suboptimal
- **Source** SafeCast.sol

Recommendation These functions could be simplified as: function safe248 (uint256 x) internal pure returns (uint248 y) { if ((y = uint248 (x)) != x) revert SafeCast__Exceeds248Bits (x); }

```
42  function safe248(uint256 x) internal pure returns (uint248) {
50  function safe240(uint256 x) internal pure returns (uint240) {
58  function safe232(uint256 x) internal pure returns (uint232) {
66  function safe224(uint256 x) internal pure returns (uint224) {
74  function safe216(uint256 x) internal pure returns (uint216) {
82  function safe208(uint256 x) internal pure returns (uint208) {
90  function safe200(uint256 x) internal pure returns (uint200) {
98  function safe192(uint256 x) internal pure returns (uint192) {
106 function safe184(uint256 x) internal pure returns (uint184) {
114 function safe176(uint256 x) internal pure returns (uint176) {
122 function safe168(uint256 x) internal pure returns (uint168) {
130 function safe160(uint256 x) internal pure returns (uint160) {
138 function safe152(uint256 x) internal pure returns (uint152) {
146 function safe144(uint256 x) internal pure returns (uint144) {
154 function safe136(uint256 x) internal pure returns (uint136) {
162 function safe128(uint256 x) internal pure returns (uint128) {
170 function safe120(uint256 x) internal pure returns (uint120) {
178 function safe112(uint256 x) internal pure returns (uint112) {
```

CVF-213. FIXED

- **Category** Procedural
- **Source** ILBToken.sol

Description Specifying an unbounded from the uppoer side version range is a bad practice, as it is impossible to guarantee compatibility with future major releases.

Recommendation Consider specifying as "^0.8.0'. Also relevant for the next files: ILBRouter.sol, IPendingOwnable.sol, IWAVAX.sol.

```
3 pragma solidity >=0.8.9;
```

CVF-215. INFO

- **Category** Suboptimal
- **Source** ILBToken.sol

Recommendation It would be more efficient to use a single array of structs with two fields, rather than two parallel arrays.

Client Comment *ERC1155 uses _ids and _values, we prefer to keep it that way*

```
12 uint256[] ids,
   uint256[] amounts
```

CVF-216. FIXED

- **Category** Suboptimal
- **Source** ILBToken.sol

Description Decimals is not a part of ERC1155, check this https://docs.openzeppelin.com/contracts/3.x/erc1155 "unlike ERC20, ERC1155 lacks a decimals field, since each token is distinct and cannot be partitioned.". DeFi ecosystem does not support decimals for ERC1155.

Recommendation Consider removing this.

```
22 function decimals() external view returns (uint8);
```

CVF-217. FIXED

- **Category** Documentation
- **Source** ILBToken.sol

Description The semantics of these functions is unclear.

Recommendation Consider documenting.

```
26 function userPositionAt(address _account, uint256 _index) external
      ↪ view returns (uint256);
```

```
28 function userPositionNb(address _account) external view returns (
      ↪ uint256);
```

CVF-218. INFO

- **Category** Suboptimal
- **Source** ILBToken.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

Client Comment *ERC1155 uses _ids and _values, we prefer to keep it that way*

```
39 uint256[] memory id,
40 uint256[] memory amount
```

CVF-219. FIXED

- **Category** Procedural
- **Source** ILBRouter.sol

Description The file imports itself. This is weird.

Recommendation Consider removing this import.

```
6 import "./ILBRouter.sol";
```

CVF-220. FIXED

- **Category** Procedural
- **Source** ILBRouter.sol

Description In others files you use @dev for such struct comments. Better to be consistent.

```
11 /// - The liquidity parameters, such as:
```

CVF-221. INFO

- **Category** Suboptimal
- **Source** ILBRouter.sol

Recommendation It would be simpler to just specify the minimum and the maximum allowed IDs.

```
34 uint256 activeIdDesired;
   uint256 idSlippage;
```

CVF-222. INFO

- **Category** Suboptimal
- **Source** ILBRouter.sol

Recommendation It would be more efficient to have a single array or structs with three fields, rather than three parallel array.

Client Comment *Since LBToken uses _ids and _values, we can't use it when removing liquidity, and we prefer to stay consistent in adding liquidity too*

```
36 int256[] deltaIds;
   uint256[] distributionX;
   uint256[] distributionY;
```

CVF-223. INFO

- **Category** Documentation
- **Source** ILBRouter.sol

Description A bin ID corresponds to a price range rather than a single price.

Recommendation Consider explaining what exact price is returned by this function for a bin ID.

Client Comment *Within a bin, the price is always the same*

```
51 function getPriceFromId(ILBPair LBPair, uint24 id) external view
      ↪ returns (uint256);
```

CVF-224. FIXED

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

Recommendation This function should return the created pair.

```
65 function createLBPair(
```

CVF-225. FIXED

- **Category** Suboptimal
- **Source** ILBRouter.sol

Recommendation These functions should emit some events and these events should be declared in this interface.

```
65  function createLBPair(
72  function addLiquidity(LiquidityParameters memory liquidityParameters
        ↪ ) external;
74  function addLiquidityAVAX(LiquidityParameters memory
        ↪ liquidityParameters) external payable;
76  function removeLiquidity(
88  function removeLiquidityAVAX(
99  function swapExactTokensForTokens(
108 function swapExactTokensForAVAX(
117 function swapExactAVAXForTokens(
125 function swapTokensForExactTokens(
134 function swapTokensForExactAVAX(
143 function swapAVAXForExactTokens(
151 function swapExactTokensForTokensSupportingFeeOnTransferTokens(
160 function swapExactTokensForAVAXSupportingFeeOnTransferTokens(
169 function swapExactAVAXForTokensSupportingFeeOnTransferTokens(
177 function sweep(
```

CVF-226. FIXED

- **Category** Suboptimal
- **Source** ILBRouter.sol

Recommendation These functions should return the liquidity amounts minted.

```
72 function addLiquidity(LiquidityParameters memory liquidityParameters
   ↪ ) external;

74 function addLiquidityAVAX(LiquidityParameters memory
   ↪ liquidityParameters) external payable;
```

CVF-227. FIXED

- **Category** Suboptimal
- **Source** ILBRouter.sol

Recommendation These functions should return the token amounts retrieved.

```
76 function removeLiquidity(

88 function removeLiquidityAVAX(
```

CVF-228. INFO

- **Category** Suboptimal
- **Source** ILBRouter.sol

Recommendation It would be more efficient to pass a single array of structs with two fields rather than two parallel arrays. This would also make the length check unnecessary.

Client Comment *tokenPath and pairVersions don't have the same length*

```
82  uint256[] memory ids,
    uint256[] memory amounts,

93  uint256[] memory ids,
    uint256[] memory amounts,

102 uint256[] memory pairVersions,
    IERC20[] memory tokenPath,

111 uint256[] memory pairVersions,
    IERC20[] memory tokenPath,

119 uint256[] memory pairVersions,
120 IERC20[] memory tokenPath,

128 uint256[] memory pairVersions,
    IERC20[] memory tokenPath,

137 uint256[] memory pairVersions,
    IERC20[] memory tokenPath,

145 uint256[] memory pairVersions,
    IERC20[] memory tokenPath,

154 uint256[] memory pairVersions,
    IERC20[] memory tokenPath,

163 uint256[] memory pairVersions,
    IERC20[] memory tokenPath,

171 uint256[] memory pairVersions,
    IERC20[] memory tokenPath,
```

CVF-229. FIXED

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

Recommendation The type of these arguments should be “address payable”.

```
95  address to,
```

```
113 address to,
```

```
139 address to,
```

```
165 address to,
```

CVF-230. FIXED

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

Recommendation These functions should return the actual output amount.

```
99  function swapExactTokensForTokens(
```

```
108 function swapExactTokensForAVAX(
```

```
117 function swapExactAVAXForTokens(
```

```
151 function swapExactTokensForTokensSupportingFeeOnTransferTokens(
```

```
160 function swapExactTokensForAVAXSupportingFeeOnTransferTokens(
```

```
169 function swapExactAVAXForTokensSupportingFeeOnTransferTokens(
```

CVF-231. FIXED

- **Category** Suboptimal
- **Source** ILBRouter.sol

Recommendation These functions should return the actual input amounts.

```
125 function swapTokensForExactTokens(
```

```
134 function swapTokensForExactAVAX(
```

```
143 function swapAVAXForExactTokens(
```

CVF-232. INFO

- **Category** Suboptimal
- **Source** ILBRouter.sol

Description This function is usable only by the owner.

Recommendation Consider removing its declaration from this interface.

```
177 function sweep(
```

CVF-233. FIXED

- **Category** Procedural
- **Source** ILBPair.sol

Recommendation Consider specifying as “^0.8.0”.

```
3 pragma solidity >=0.8.9;
```

CVF-234. INFO

- **Category** Suboptimal
- **Source** ILBPair.sol

Description 112 bit for a token amount doesn’t look sufficient in general. Usually DeFi allow at least 128 bit for amounts.

Recommendation Consider using more bits for token amounts or packing amounts somehow.

```
15 uint112 reserveX;
   uint112 reserveY;
```

CVF-235. INFO

- **Category** Documentation
- **Source** ILBPair.sol

Description These fields are not documented and their semantics is unclear.

Recommendation Consider documenting.

```
17 uint256 accTokenXPerShare;
   uint256 accTokenYPerShare;
```

CVF-236. INFO

- **Category** Documentation
- **Source** ILBPair.sol

Description It is unclear what the current ID is, and the comment doesn't help.

Recommendation Consider elaborating a bit more one this.

```
23 /// - activeId: The current id used for swaps, this is also linked
      ↪ with the price
```

```
37     uint24 activeId;
```

CVF-237. INFO

- **Category** Documentation
- **Source** ILBPair.sol

Description It is unclear what an oracle ID is, and the comment doesn't help.

Recommendation Consider elaborating a bit more on this.

```
31 /// - oracleId: The current id of the oracle
```

```
44     uint24 oracleId;
```

CVF-238. INFO

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

Recommendation In some places you use uint112 for reserves, in others uint136, better to keep consistency.

Client Comment *uint136 are for total reserves as there is a maximum of 2**24 bins of 2**112 tokens*

```
38 uint136 reserveX;
   uint136 reserveY;
```

CVF-239. INFO

- **Category** Suboptimal
- **Source** ILBPair.sol

Recommendation Since reserves are uint112, you can use smaller uint here to save storage space.

```
53 uint256 debtX;
   uint256 debtY;
```

CVF-240. FIXED

- **Category** Procedural
- **Source** ILBPair.sol

Recommendation No comments here, however all other structures have comments about every attribute. Better to be consistent.

```
65 struct MintInfo {
       uint256 amountXIn;
       uint256 amountYIn;
       uint256 amountXAddedToPair;
       uint256 amountYAddedToPair;
70     uint256 totalDistributionX;
       uint256 totalDistributionY;
       uint256 id;
       uint256 amount;
   }
```

CVF-241. INFO

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

Recommendation Most of DEXes names such attributes as token0 and token1, better to follow this practice.

Client Comment *We use X and Y because Y is the quote asset, while 1 and 0 are connoted to be ordered*

```
76 function tokenX() external view returns (IERC20);

   function tokenY() external view returns (IERC20);
```

CVF-242. INFO

- **Category** Documentation
- **Source** ILBPair.sol

Description The semantics of these returned values is unclear.

Recommendation Consider documenting.

```
105 uint256 min,
    uint256 max
```

CVF-243. INFO

- **Category** Documentation
- **Source** ILBPair.sol

Description The semantics of these returned values is unclear.

Recommendation Consider documenting.

```
113 uint256 cumulativeId,
    uint256 cumulativeAccumulator,
    uint256 cumulativeBinCrossed
```

CVF-244. INFO

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

Description The semantics of the "sendTokenY" argument is unclear. This argument seems to specify the search direction.

Recommendation Consider renaming it.

Client Comment *We believe that it's easier to understand like that than using the search direction*

```
120 function findFirstNonEmptyBinId(uint24 id, bool sentTokenY) external
       ↪  view returns (uint256);
```

CVF-245. INFO

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

Description The function name is confusing as it actually searches for the next non-empty bin after (or before) the given bit.

Recommendation Consider renaming.

```
120 function findFirstNonEmptyBinId(uint24 id, bool sentTokenY) external
       ↪  view returns (uint256);
```

CVF-246. FIXED

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

Description The semantics of the returned values is unclear.

Recommendation Consider giving descriptive names to the returned values and/or adding a documentation comment.

126
```
function swap(bool sentTokenY, address to) external returns (uint256
    ↪ , uint256);
```

CVF-247. FIXED

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

Recommendation These functions should emit some events, and these events should be declared in this function.

126
```
function swap(bool sentTokenY, address to) external returns (uint256
    ↪ , uint256);
```

128
```
function flashLoan(
```

135
```
function mint(
```

142
```
function burn(
```

148
```
function increaseOracleLength(uint16 _nb) external;
```

150
```
function collectFees(address _account, uint256[] memory _ids)
    ↪ external;
```

152
```
function collectProtocolFees() external;
```

154
```
function setFeesParameters(bytes32 packedFeeParameters) external;
```

CVF-248. FIXED

- **Category** Suboptimal
- **Source** ILBPair.sol

Recommendation This function should also return the liquidity amounts minted for each bin.

135
```
function mint(
```

CVF-249. FIXED

- **Category** Suboptimal
- **Source** ILBPair.sol

Recommendation It would be more efficient to pass a single array of structs with three fields, rather than three parallel arrays. Such approach would also make the length check unnecessary.

```
136 uint256[] memory _ids,
    uint256[] memory _distributionX,
    uint256[] memory _distributionY,
```

CVF-250. FIXED

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

Description The semantics of the returned values is unclear.

Recommendation Consider giving descriptive names to the returned values and/or adding a documentation comment.

```
140 ) external returns (uint256, uint256);
```

CVF-251. FIXED

- **Category** Suboptimal
- **Source** ILBPair.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather then two parallel arrays. Such approach would also make the length check unnecessary.

```
143 uint256[] memory ids,
    uint256[] memory _amounts,
```

CVF-252. FIXED

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

Description The semantics of the returned values is unclear.

Recommendation Consider giving descriptive names to the returned values and/or adding a documentation comment.

```
146 ) external returns (uint256, uint256);
```

CVF-253. FIXED

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

Description The semantics of the function is unclear.

Recommendation Consider documenting.

Client Comment *The body changed*

```
148 function increaseOracleLength(uint16 _nb) external;
```

CVF-254. FIXED

- **Category** Suboptimal
- **Source** ILBPair.sol

Recommendation This function should return the collected amounts.

```
150 function collectFees(address _account, uint256[] memory _ids)
      ↪ external;
```

CVF-255. FIXED

- **Category** Suboptimal
- **Source** ILBPair.sol

Recommendation This function should return the collected amounts.

```
152 function collectProtocolFees() external;
```

CVF-256. FIXED

- **Category** Suboptimal
- **Source** IPendingOwnable.sol

Recommendation These functions should emit some events and these events should be declared in this interface.

Client Comment *Fixed somewhere*

```
10 function setPendingOwner(address pendingOwner) external;
```

```
12 function revokePendingOwner() external;
```

```
14 function becomeOwner() external;
```

```
16 function renounceOwnership() external;
```

CVF-257. INFO

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

Description The semantics of the argument is unclear.

Recommendation Consider giving the argument a descriptive name and/or adding a documentation comment.

```
10 function withdraw(uint256) external;
```

CVF-258. FIXED

- **Category** Procedural
- **Source** ILBFlashLoanCallback.sol

Description Specifying an unbounded from the upper side version range is a bad practice as it is impossible to guarantee compatibility with future major releases.

Recommendation Consider specifying as “^0.8.0”.

```
3 pragma solidity >=0.8.9;
```

CVF-259. INFO

- **Category** Procedural
- **Source** ILBFlashLoanCallback.sol

Recommendation Function names usually start with lower case letters.

Client Comment *We agreed that acronyms should be in caps letter*

```
6 function LBFlashLoanCallback(
```

CVF-260. FIXED

- **Category** Procedural
- **Source** ILBFactory.sol

Description Specifying an unbounded from the upper side version range is a bad practice as it is impossible to guarantee compatibility with future major releases.

Recommendation Consider specifying as “^0.8.0”.

```
3 pragma solidity >=0.8.9;
```

CVF-261. INFO

- **Category** Documentation
- **Source** ILBFactory.sol

Description The number format of the returned values in unclear.

Recommendation Consider documenting.

```
34 function MIN_FEE() external pure returns (uint256);
```

```
36 function MAX_FEE() external pure returns (uint256);
```

```
42 function MAX_PROTOCOL_SHARE() external pure returns (uint256);
```

CVF-262. FIXED

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

Description Despite the name, this function returns only one LB pair.

Recommendation Consider renaming.

```
50 function allLBPairs(uint256 id) external returns (ILBPair);
```

CVF-263. FIXED

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

Description Despite the name, this function returns the number of LB pairs.

Recommendation Consider renaming.

```
52 function allPairsLength() external view returns (uint256);
```

CVF-264. FIXED

- **Category** Documentation
- **Source** ILBFactory.sol

Description The semantics of this function is unclear.

Recommendation Consider documenting.

```
60 function setFactoryHelper() external;
```

CVF-265. INFO

- **Category** Suboptimal
- **Source** ILBFactory.sol

Recommendation These functions should emit some events and these events should be declared in this interface.

```
60  function setFactoryHelper() external;

62  function createLBPair(

69  function setFeeRecipient(address feeRecipient) external;

71  function setLBPairBlacklist(

78  function setPreset(

90  function removePreset(uint16 binStep) external;

113 function setFeeParametersOnPair(
```

CVF-266. INFO

- **Category** Procedural
- **Source** ILBFactory.sol

Description The types of set and retrieved values don't match.

Recommendation Consider making them consistent.

```
80  uint8 _baseFactor,
    uint16 _filterPeriod,
    uint16 _decayPeriod,
    uint8 _reductionFactor,
    uint8 _variableFeeControl,
    uint8 _protocolShare,
    uint72 _maxAccumulator,
    uint8 _sampleLifetime
```

```
96      uint256 baseFactor,
        uint256 filterPeriod,
        uint256 decayPeriod,
        uint256 reductionFactor,
100     uint256 variableFeeControl,
        uint256 protocolShare,
        uint256 maxAccumulator,
        uint256 sampleLifetime
```

CVF-267. INFO

- **Category** Suboptimal
- **Source** ILBFactory.sol

Description These functions don't scale and there could be too many bit steps to be returned in one transaction.

Recommendation Consider implementing an ability to retrieve only a portion of the result and/or checking individual bit steps for whether they are available.

Client Comment *There will never be more than 100 bin steps, same for presets as the binStep is unique in presets*

```
106 function getAvailablePresetsBinStep() external view returns (uint256
        ↪ [] memory presetsBinStep);
```

```
108 function getAvailableLBPairsBinStep(IERC20 _tokenX, IERC20 _tokenY)
```

CVF-268. FIXED

- **Category** Procedural
- **Source** ILBFactoryHelper.sol

Description Specifying an unbounded from the upper side version range is a bad practice, as it is impossible to guarantee compatibility with future major releases.

Recommendation Consider specifying as "^0.8.0".

```
3 pragma solidity >=0.8.9;
```

CVF-269. FIXED

- **Category** Suboptimal
- **Source** ILBFactoryHelper.sol

Recommendation This function should emit some event, and this event should be declared in this interface.

```
13 function createLBPair(
```

CVF-270. INFO

- **Category** Documentation
- **Source** ILBFactoryHelper.sol

Description The structure of this argument is unclear.

Recommendation Consider documenting.

Client Comment *It's documented in the main contract*

```
19 bytes32 packedFeeParameters
```

CVF-271. FIXED

- **Category** Procedural
- **Source** IJoePair.sol

Description Specifying an unbounded from the upper side version range is a bad practice as it is impossible to guarantee compatibility with the future major releases.

Recommendation Consider specifying like “^0.5.0 || ^0.6.0 || ^0.7.0 || ^0.8.0”.

```
3 pragma solidity >=0.5.0;
```

CVF-272. INFO

- **Category** Procedural
- **Source** IJoePair.sol

Description These events and functions are declared by ERC20 standard.

Recommendation Consider removing them from here and inheriting this interface from "IERC20Metadata".

```
6  event Approval(address indexed owner, address indexed spender,
       ↪ uint256 value);
   event Transfer(address indexed from, address indexed to, uint256
       ↪ value);

9  function name() external pure returns (string memory);

11 function symbol() external pure returns (string memory);

13 function decimals() external pure returns (uint8);

15 function totalSupply() external view returns (uint256);

17 function balanceOf(address owner) external view returns (uint256);

19 function allowance(address owner, address spender) external view
       ↪ returns (uint256);

21 function approve(address spender, uint256 value) external returns (
       ↪ bool);

23 function transfer(address to, uint256 value) external returns (bool)
       ↪ ;

25 function transferFrom(
       address from,
       address to,
       uint256 value
   ) external returns (bool);
```

CVF-273. INFO

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

Description Despite the name this function returns a signle nonce.

Recommendation Consider renaming.

```
35 function nonces(address owner) external view returns (uint256);
```

CVF-274. INFO

- **Category** Procedural
- **Source** IJoePair.sol

Description In contrast to other similar events, this event doesn't have the "to" parameter, which means that the sender and the beneficiary is always the same.

Recommendation Consider adding the "to" parameter for completeness even if in the current version different beneficiary is not supported yet.

```
47 event Mint(address indexed sender, uint256 amount0, uint256 amount1)
       ↪ ;
```

CVF-275. INFO

- **Category** Suboptimal
- **Source** IJoePair.sol

Description Separate "in" and "out" amounts for both tokens look redundant.

Recommendation Consider having one amount for each token and a separate swap direction.

```
51 uint256 amount0In,
   uint256 amount1In,
   uint256 amount0Out,
   uint256 amount1Out,
```

CVF-276. INFO

- **Category** Documentation
- **Source** IJoePair.sol

Description The semantics of this event is unclear.

Recommendation Consider documenting.

```
57 event Sync(uint112 reserve0, uint112 reserve1);
```

CVF-277. INFO

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

Recommendation The return type should be “IJoeFactory”.

```
61 function factory() external view returns (address);
```

CVF-278. INFO

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

Recommendation The return type should be “IERC20”.

```
63 function token0() external view returns (address);
```

```
65 function token1() external view returns (address);
```

CVF-279. INFO

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

Description The semantics of this return value is unclear.

Recommendation Consider giving a descriptive name to the return value and/or adding a documenataion comment.

```
73 uint32 blockTimestampLast
```

CVF-280. INFO

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

Description The semantics of the returned values is unclear.

Recommendation Consider giving descriptive names to the return values and/or adding documenataion comments.

```
76 function price0CumulativeLast() external view returns (uint256);
```

```
78 function price1CumulativeLast() external view returns (uint256);
```

```
80 function kLast() external view returns (uint256);
```

CVF-281. INFO

- **Category** Documentation
- **Source** IJoePair.sol

Description The semantics of these functions is unclear.

Recommendation Consider adding documentation comments.

```
93 function skim(address to) external;
```

```
95 function sync() external;
```

CVF-282. INFO

- **Category** Documentation
- **Source** IJoePair.sol

Description The semantics of the arguments is unclear.

Recommendation Consider giving descriptive names to the arguments and/or adding a documentation comment.

```
97 function initialize(address, address) external;
```

CVF-283. FIXED

- **Category** Procedural
- **Source** IJoeFactory.sol

Description Specifying an unbounded from the upper side version range is a bad practice as it is impossible to guarantee compatibility with the future major releases.

Recommendation Consider specifying like "^0.5.0 || ^0.6.0 || ^0.7.0 || ^0.8.0".

```
3 pragma solidity >=0.5.0;
```

CVF-284. INFO

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

Recommendation This type of the token parameters should be "IERC20". The type of the "pair" parameter should be "IJoePair".

```
6 event PairCreated(address indexed token0, address indexed token1,
    ↪ address pair, uint256);
```

CVF-285. INFO

- **Category** Documentation
- **Source** IJoeFactory.sol

Description The semantics of the last parameter is unclear.

Recommendation Consider giving the last parameter a descriptive name and/or adding a documentation comment.

```
6 event PairCreated(address indexed token0, address indexed token1,
    ↪ address pair, uint256);
```

CVF-286. INFO

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

Recommendation The arguments type should be “IERC20”. The return type should be “IJoePair”.

```
14 function getPair(address tokenA, address tokenB) external view
    ↪ returns (address pair);
```

CVF-287. INFO

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

Description Despite the name, this function returns only one pair.

Recommendation Consider renaming.

```
16 function allPairs(uint256) external view returns (address pair);
```

CVF-288. INFO

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

Description The semantics of the argument is unclear.

Recommendation Consider giving a descriptive name to the argument and/or adding a documentation comment.

```
16 function allPairs(uint256) external view returns (address pair);
```

CVF-289. INFO

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

Recommendation The return type should be “IJoePair”.

```
16 function allPairs(uint256) external view returns (address pair);
```

CVF-290. INFO

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

Recommendation Better name would be “pairsCount”.

```
18 function allPairsLength() external view returns (uint256);
```

CVF-291. INFO

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

Recommendation The arguments type should be “IERC20”. The return type should be “IJoePair”.

```
20 function createPair(address tokenA, address tokenB) external returns
   ↪  (address pair);
```

CVF-292. INFO

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

Recommendation These functions should emit some events and these events should be declared in this interface. Alternatively, consider splitting this interface into “user” and “administration” parts.

```
22 function setFeeTo(address) external;

24 function setFeeToSetter(address) external;

26 function setMigrator(address) external;
```

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