HALBORN

Unlimited Network – Unlimited Leverage

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **January 23rd, 2023 - April 11th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	01/23/2023	Roberto Reigada
0.2	Document Updates	03/03/2023	Roberto Reigada
0.3	Draft Review	03/03/2023	Gokberk Gulgun
0.4	Draft Review	03/03/2023	Gabi Urrutia
1.0	Remediation Plan	03/28/2023	Roberto Reigada
1.1	Remediation Plan Review	03/28/2023	Gokberk Gulgun
1.2	Remediation Plan Review	03/28/2023	Gabi Urrutia
2.0	New Scope Document Updates	04/10/2023	Ataberk Yavuzer
2.1	New Scope Document Review	04/12/2023	Piotr Cielas
2.2	New Scope Document Review	04/12/2023	Gabi Urrutia
2.3	New Scope Remediation Plan	04/25/2023	Ataberk Yavuzer
2.3	New Scope Remediation Plan Review	04/25/2023	Piotr Cielas
2.4	New Scope Remediation Plan Review	04/25/2023	Gabi Urrutia
2.5	Final Commit Report Update	05/19/2023	Ataberk Yavuzer

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com
Ataberk Yavuzer	Halborn	Ataberk.Yavuzer@halborn.com
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Unlimited Leverage is a synthetic leverage trading platform, where users can trade their favorite cryptocurrencies synthetically with zero price impact, minimal slippage, and up to 100x leverage.

Unlimited Network engaged Halborn to conduct a security audit on their smart contracts beginning on January 23rd, 2023 and ending on April 11th, 2023. The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided **5 weeks** for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

After the first audit was completed, a new commit hash was sent to Halborn by the Unlimited Network team. A **5-day** audit was carried out between April 5th and April 10th.

The purpose of the audits is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified some issues that were mostly addressed by the Unlimited Network team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the contracts' solidity code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing with custom scripts. (Foundry).
- Static Analysis of security for scoped contract, and imported functions manually.
- Testnet deployment (Ganache).

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.

3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

1. IN-SCOPE:

The security assessment was scoped to the following smart contracts on the halborn-audit branch:

- FeeManager.sol
- LiquidityPool.sol
- LiquidityPoolAdapter.sol
- ChainlinkUsdPriceFeed.sol
- ChainlinkUsdPriceFeedPrevious.sol
- PriceFeedAdapter.sol
- PriceFeedAggregator.sol
- UnlimitedPriceFeed.sol
- UnlimitedPriceFeedAdapter.sol
- Controller.sol
- UnlimitedOwner.sol
- TradeManagerOrders.sol
- TradePair.sol
- TradePairHelper.sol
- UserManager.sol

Commit ID : b29d5285be299e5a55942ce4e8c4eef1595ad1ed

2. REMEDIATION PR/COMMITS:

Fix Commit ID : 41eda38f650af8b837400bba48316c776a33c61f

3. SECOND AUDIT COMMITS:

Second Audit Commit IDs:

1. PR #121
2. 41c6ac1003daa056e9ce7c6c2f5cfa47b8c0ac0b
3. f50f1289f4fcb68f9aac081d08d387d3f53ab165

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	4	8	6

LIKELIHOOD

IMPACT

(HAL-04) (HAL-05)	(HAL-02) (HAL-03)			
(HAL-06) (HAL-09) (HAL-13)				
(HAL-10)	(HAL-07)			(HAL-01)
	(HAL-11) (HAL-12)			
(HAL-14) (HAL-15) (HAL-16) (HAL-17) (HAL-18) (HAL-19)			(HAL-08)	

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL01) PRECISION LOSS IN PARTIALLYCLOSEPOSITION FUNCTION WILL BLOCK THE LAST USER FROM CLOSING THE POSITION	High	SOLVED - 05/19/2023
(HAL02) OPENPOSITIONVIASIGNATURE DOES NOT CHECK THAT THE POSITION IS NOT LIQUIDATABLE	Medium	SOLVED - 05/19/2023
(HAL03) USE OF DEPRECATED CHAINLINK FUNCTION: LATESTANSWER	Medium	SOLVED - 05/19/2023
(HAL04) UNLIMITEDPRICEFEED IS VULNERABLE TO CROSSCHAIN SIGNATURE REPLAY ATTACKS	Medium	SOLVED - 05/19/2023
(HAL05) CENTRALIZATION RISK: ORDEREXECUTOR COULD CLOSE POSITIONS IN HIS BENEFIT BEFORE EVERY PRICE UPDATE	Medium	RISK ACCEPTED
(HAL06) FIRST LIQUIDITYPOOL DEPOSITOR COULD BE FRONT-RUN	Low	SOLVED - 05/19/2023
(HAL07) MULTIPLE FUNCTIONS DO NOT CHECK THAT THE NEW MARGIN IS HIGHER THAN THE MINIMUM MARGIN	Low	SOLVED - 05/19/2023
(HAL08) ADDING MARGIN TO A POSITION MAY RESULT IN A NEGATIVE LIQUIDATION PRICE	Low	SOLVED - 05/19/2023
(HAL09) LACK OF A DOUBLE-STEP TRANSFEROWNERSHIP PATTERN	Low	SOLVED - 05/19/2023
(HAL10) LACK OF DISABLEINITIALIZERS CALL TO PREVENT UNINITIALIZED CONTRACTS	Low	SOLVED - 03/28/2023
(HAL11) UNLIMITEDPRICEADAPTER LACKS THE ONLYOWNER MODIFIER ON INITIALIZER FUNCTION	Low	SOLVED - 04/20/2023
(HAL12) UPGRADEABLE CONTRACTS LACK RESERVED SPACE FOR FUTURE UPGRADES	Low	SOLVED - 05/19/2023
(HAL13) POSSIBLE DOS WITH BLOCK GAS LIMIT	Low	SOLVED - 05/19/2023

(HAL14) INCOMPATIBILITY WITH TRANSFER-ON-FEE OR DEFLATIONARY TOKENS	Informational	ACKNOWLEDGED
(HAL15) USE ++I INSTEAD OF I++ IN LOOPS FOR LOWER GAS COSTS	Informational	SOLVED - 05/19/2023
(HAL16) UNNEEDED INITIALIZATION OF INTEGER VARIABLES TO 0	Informational	SOLVED - 05/19/2023
(HAL17) BACKEND ADDRESS WOULD HIGHLY BENEFIT OF TRAILING ZERO BYTES	Informational	ACKNOWLEDGED
(HAL18) DETAILSOFPOSITION FUNCTION CAN BE OPTIMIZED	Informational	SOLVED - 05/19/2023
(HAL19) REVERT MESSAGES LONGER THAN 32 BYTES COST EXTRA GAS	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) PRECISION LOSS IN PARTIALLYCLOSEPOSITION FUNCTION WILL BLOCK THE LAST USER FROM CLOSING THE POSITION - HIGH

Description:

The partiallyClosePositionViaSignature() function call has the following flow:

1. TradeManagerOrders.partiallyClosePositionViaSignature()
2. TradeManagerOrders._partiallyClosePosition()
3. TradePair.partiallyClosePosition()
4. TradePair._partiallyClosePosition()
5. PositionMaths.partiallyClose()

Listing 1: PositionMaths.sol (Lines 438,452,453,454)

```
419 /**
420  * @dev Partially closing works as follows:
421  *
422  * 1. Sell a share of the position, and use the proceeds to either
 ↳ :
423  * 2.a) Get a payout and by this, leave the leverage as it is
424  * 2.b) "Buy" new margin and by this decrease the leverage
425  * 2.c) a mixture of 2.a) and 2.b)
426  */
427 function _partiallyClose(Position storage self, int256
 ↳ currentPrice, uint256 closeProportion)
428     internal
429     returns (int256)
430 {
431     require(
432         closeProportion < PERCENTAGE_MULTIPLIER,
433         "PositionMaths::_partiallyClose: cannot partially close
 ↳ full position"
434     );
435
```

```
436     Position memory delta;
437     // Close a proportional share of the position
438     delta.margin = self._lastNetMargin() * closeProportion /
 ↳ PERCENTAGE_MULTIPLIER;
439     delta.volume = self.volume * closeProportion /
 ↳ PERCENTAGE_MULTIPLIER;
440     delta.assetAmount = self.assetAmount * closeProportion /
 ↳ PERCENTAGE_MULTIPLIER;
441
442     // The realized PnL is the change in volume minus the price of
 ↳  the changes in size at LONG
443     // And the inverse of that at SHORT
444     // @dev At a long position, the delta of size is sold to give
 ↳ back the volume
445     // @dev At a short position, the volume delta is used, to "buy
 ↳ " the change of size (and give it back)
446     int256 priceOfSizeDelta = currentPrice * int256(delta.
 ↳ assetAmount) / int256(10 ** self.assetDecimals); //
 ↳ priceOfAssetAmountDelta
447     int256 realizedPnL = (priceOfSizeDelta - int256(delta.volume))
 ↳  * self._shortMultiplier();
448
449     int256 payout = int256(delta.margin) + realizedPnL;
450
451     // change storage values
452     self.margin -= delta.margin;
453     self.volume -= delta.volume;
454     self.assetAmount -= delta.assetAmount;
455
456     // Return payout for further calculations
457     return payout;
458 }
```

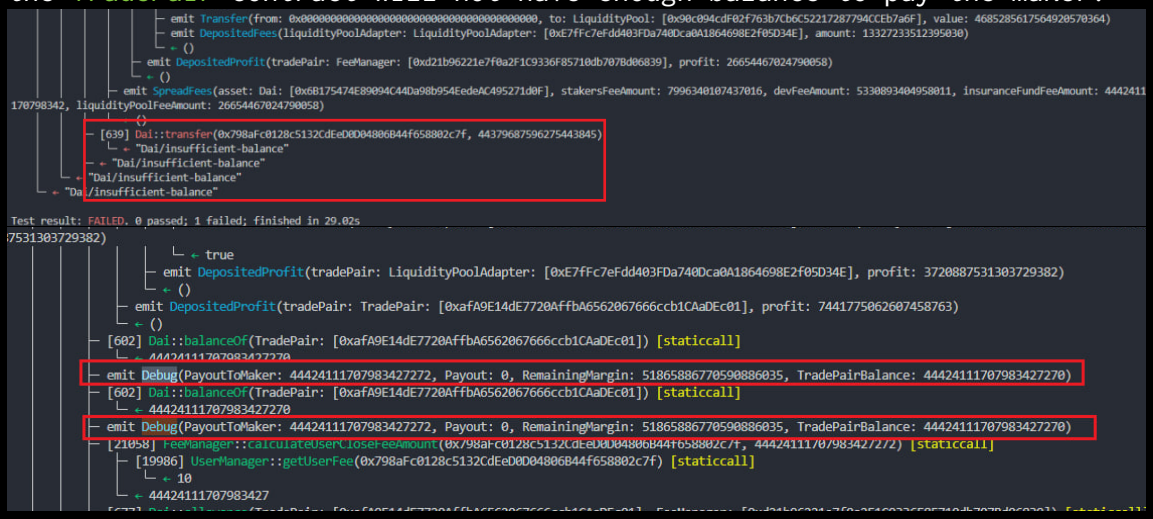
In the line 438 of the PositionMaths library, the following operation is performed:
`delta.margin = self._lastNetMargin()*closeProportion/PERCENTAGE_MULTIPLIER`

In that division, some precision is lost and the `delta.margin` value is then subtracted from the storage: `self.margin -= delta.margin`.

This precision loss here will cause 2 different problems.

PROBLEM #1:

The last user in the system will never be able to close his position as the TradePair contract will not have enough balance to pay the maker:



Listing 2: TradePair.sol (Lines 279,283,285)

```
259 function _closePosition(uint256 positionId_) private {
260     Position storage position = positions[positionId_];
261
262     // Clear Buffer
263     (uint256 remainingMargin, uint256 remainingBufferFee, uint256
 ↳ requestLoss) = _clearBuffer(position, false);
264
265     // Get the payout to the maker
266     uint256 payoutToMaker = _getPayoutToMaker(position);
267
268     // update aggregated values
269     positionStats.removeTotalCount(position.margin, position.
 ↳ volume, position.assetAmount, position.isShort);
270
271     int256 protocolPnL = int256(remainingMargin) - int256(
 ↳ payoutToMaker) - int256(requestLoss);
272
273     // fee manager receives the remaining fees
274     _depositBorrowFees(remainingBufferFee);
275
276     uint256 payout = _registerProtocolPnL(protocolPnL);
277
278     // Make sure the payout to maker does not exceed the
 ↳ collateral for this position made up of the remaining margin and
 ↳ the (possible) received loss payout
```

```
279     emit Debug(payoutToMaker, payout, remainingMargin, collateral.
 ↳ balanceOf(address(this)));
280     if (payoutToMaker > payout + remainingMargin) {
281         payoutToMaker = payout + remainingMargin;
282     }
283     emit Debug(payoutToMaker, payout, remainingMargin, collateral.
 ↳ balanceOf(address(this)));
284     if (payoutToMaker > 0) {
285         _payoutToMaker(position.owner, int256(payoutToMaker),
 ↳ positionId_);
286     }
287
288     emit RealizedPnL(position.owner, positionId_,
 ↳ _getCurrentNetPnL(position));
289
290     emit ClosedPosition(positionId_, _getCurrentPrice(position.
 ↳ isShort, true));
291
292     // Finally delete position
293     _deletePosition(positionId_);
294 }
```

PROBLEM #2:

At a position closure, when _registerProtocolPnL() is called the TradePair smart contract will not have enough balance to transfer protocolPnL_ to the LiquidityPoolAdapter:

```
    ├─ [562] ERC20::balanceOf(TradePair: [0x082426eCDb60B017a5749505342 7fbf86Ac8d6a2]) [staticcall]
    │   └─ ← 587616469974126726821
    ├─ emit Debug2(ProtocolPNL: 587616469974126726822, TradePairBalance: 587616469974126726821)
    ├─ [860] ERC20::transfer(LiquidityPoolAdapter: [0x6d31e8BbeB0decDF4f5565895D2c2B32A86A97b7], 587616469974126726822)
    │   └─ ← "ERC20: transfer amount exceeds balance"
    └─ ← "ERC20: transfer amount exceeds balance"
└─ ← "ERC20: transfer amount exceeds balance"

st result: FAILED. 0 passed; 1 failed; finished in 219.58s
```

```
    ├─ [3860] PositionMaths::dff0c189(13da86008ba1c6922daee3e07db95305ef49ebced9f5467a0b8613fcc6b343e3000000000000
0000000000000000001a7daf1000000000000000000000000000000000000000000000000000000000000000000000000000000000) [delegatecall]
    │   └─ ← 0xffffffffffffffffffffffffffffffffffffffffffffffffb2697b3e0d3ccf41f
    ├─ [562] ERC20::balanceOf(TradePair: [0x082426eCDb60B017a5749505342 7fbf86Ac8d6a2]) [staticcall]
    │   └─ ← 17784595720470876855 61
    ├─ emit Debug2(ProtocolPNL: 17857464865309502290 3, TradePairBalance: 17784595720470876855 61)
    ├─ [23038] ERC20::transfer(LiquidityPoolAdapter: [0x6d31e8BbeB0decDF4f5565895D2c2B32A86A97b7], 17857464865309
    │   ├─ emit Transfer(from: TradePair: [0x082426eCDb60B017a5749505342 7fbf86Ac8d6a2], to: LiquidityPoolAdapter:
    │   └─ ← true
    ├─ [67304] LiquidityPoolAdapter::depositProfit(178574648653095022903)
    │   ├─ [562] Controller::isTradePair(TradePair: [0x082426eCDb60B017a5749505342 7fbf86Ac8d6a2]) [staticcall]
```

Listing 3: TradePair.sol (Line 724)

```
716 /**
717  * @notice Registers profit or loss at liquidity pool adapter
718  * @param protocolPnL_ Profit or loss of protocol
719  * @return payout Payout received from the liquidity pool adapter
720  */
721 function _registerProtocolPnL(int256 protocolPnL_) internal
 ↳ returns (uint256 payout) {
722     if (protocolPnL_ > 0) {
723         // Profit
724         collateral.safeTransfer(address(liquidityPoolAdapter),
 ↳ uint256(protocolPnL_));
725         liquidityPoolAdapter.depositProfit(uint256(protocolPnL_));
726     } else if (protocolPnL_ < 0) {
727         // Loss
728         payout = liquidityPoolAdapter.requestLossPayout(uint256(-
 ↳ protocolPnL_));
729     }
730     // if PnL == 0, nothing happens
731 }
```

Risk Level:

Likelihood - 5
Impact - 3

Recommendation:

It is recommended to use mulDiv() from the Uniswap's FullMath library in all the divisions to avoid the precision loss.

References:
Uniswap V3's docs
Mathematical explanation

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

During the second review, it has been observed that this change was committed on PR #121 and merged into the master branch.

3.2 (HAL-02) OPENPOSITIONVIASIGNATURE DOES NOT CHECK THAT THE POSITION IS NOT LIQUIDATABLE - MEDIUM

Description:

In the TradeManagerOrders contract, the function openPositionViaSignature() is used to an open a new position in Unlimited Leverage. This function is missing the _verifyPositionsValidity(positionId_); check. As this check is missing, users may open liquidatable positions directly.

Under a wrong configuration or a high liquidation fee, this could be abused to drain all the liquidity from the liquidity pools by creating tiny liquidatable positions and then liquidating them to get the liquidation fee.

Code Location:

Listing 4: TradeManagerOrders.sol (Line 49)

```
29 /**
30  * @notice Opens a position with a signature
31  * @param order_ Order struct
32  * @param maker_ address of the maker
33  * @param signature_ signature of order_ by maker_
34  */
35 function openPositionViaSignature(
36     OpenPositionOrder calldata order_,
37     UpdateData[] calldata updateData_,
38     address maker_,
39     bytes calldata signature_
40 ) external onlyOrderExecutor onlyActiveTradePair(order_.params.
↳ tradePair) returns (uint256) {
41     _updateContracts(updateData_);
42     _processSignature(order_, maker_, signature_);
43     _verifyConstraints(
```

```
44          order_.params.tradePair, order_.constraints, order_.params
↳ .isShort ? UsePrice.MAX : UsePrice.MIN
45      );
46
47      _transferOrderReward(order_.params.tradePair, maker_, msg.
↳ sender);
48
49      uint256 positionId = _openPosition(order_.params, maker_);
50
51      sigHashToTradeId[keccak256(signature_)] = TradeId(order_.
↳ params.tradePair, uint96(positionId));
52
53      emit OpenedPositionViaSignature(order_.params.tradePair,
↳ positionId, signature_);
54
55      return positionId;
56  }
```

Listing 5: TradeManager.sol (Lines 59-61)

```
49  /**
50   * @notice Opens a position for a trading pair.
51   * @param params_ The parameters for opening a position.
52   * @param maker_ Maker of the position
53   */
54  function _openPosition(OpenPositionParams memory params_, address
↳ maker_) internal returns (uint256) {
55      ITradePair(params_.tradePair).collateral().safeTransferFrom(
↳ maker_, address(params_.tradePair), params_.margin);
56
57      userManager.setUserReferrer(maker_, params_.referrer);
58
59      uint256 id = ITradePair(params_.tradePair).openPosition(
60          maker_, params_.margin, params_.leverage, params_.isShort,
↳  params_.whitelabelAddress
61      );
62
63      emit PositionOpened(params_.tradePair, id);
64
65      return id;
66  }
```

Listing 6: TradePair.sol (Line 189)

```
170 /**
171  * @notice opens a position
172  * @param maker_ owner of the position
173  * @param margin_ the amount of collateral used as a margin
174  * @param leverage_ the target leverage, should respect
 ↳ LEVERAGE_MULTIPLIER
175  * @param isShort_ bool if the position is a short position
176  */
177 function openPosition(address maker_, uint256 margin_, uint256
 ↳ leverage_, bool isShort_, address whitelabelAddress)
178     external
179     verifyLeverage(leverage_)
180     onlyTradeManager
181     syncFeesBefore
182     checkAssetAmountLimitAfter
183     returns (uint256)
184 {
185     if (whitelabelAddress != address(0)) {
186         positionIdToWhiteLabel[nextId] = whitelabelAddress;
187     }
188
189     return _openPosition(maker_, margin_, leverage_, isShort_);
190 }
191
192 /**
193  * @dev Should have received margin from TradeManager
194  */
195 function _openPosition(address maker_, uint256 margin_, uint256
 ↳ leverage_, bool isShort_)
196     private
197     returns (uint256)
198 {
199     require(margin_ >= minMargin, "TradePair::_openPosition:
 ↳ margin must be above or equal min margin");
200
201     uint256 id = nextId;
202     nextId++;
203
204     margin_ = _deductAndTransferOpenFee(maker_, margin_, leverage_
 ↳ , id);
205
206     uint256 volume = (margin_ * leverage_) / LEVERAGE_MULTIPLIER;
207     require(volume <= volumeLimit, "TradePair::_openPosition:
```

```
 ↳ borrow limit reached");
208     _registerUserVolume(maker_, volume);
209
210     uint256 assetAmount;
211     if (isShort_) {
212         assetAmount = priceFeedAdapter.collateralToAssetMax(volume
 ↳ );
213     } else {
214         assetAmount = priceFeedAdapter.collateralToAssetMin(volume
 ↳ );
215     }
216
217     (int256 currentBorrowFeeIntegral, int256
 ↳ currentFundingFeeIntegral) = _getCurrentFeeIntegrals(isShort_);
218
219     positions[id] = Position({
220         margin: margin_,
221         volume: volume,
222         assetAmount: assetAmount,
223         pastBorrowFeeIntegral: currentBorrowFeeIntegral,
224         lastBorrowFeeAmount: 0,
225         pastFundingFeeIntegral: currentFundingFeeIntegral,
226         lastFundingFeeAmount: 0,
227         lastFeeCalculationAt: uint48(block.timestamp),
228         openedAt: uint48(block.timestamp),
229         isShort: isShort_,
230         owner: maker_,
231         assetDecimals: uint16(assetDecimals),
232         lastAlterationBlock: uint40(block.number)
233     });
234
235     userToPositionIds[maker_].push(id);
236
237     positionStats.addTotalCount(margin_, volume, assetAmount,
 ↳ isShort_);
238
239     emit OpenedPosition(maker_, id, margin_, volume, assetAmount,
 ↳ isShort_);
240
241     return id;
242 }
```

Proof of Concept:

```
test_env_OpenSmallPosition()
  ---------------------------------------------------------------------------------------------------------------------------

CALLING -> contract_tradePair0.setLiquidatorReward(10e18);

CALLING -> contract_tradePair0.setMinMargin(5e18);

CALLING -> contract_Controller.setOrderRewardOfCollateral(address(contract_Collateral), 1e18);
  orderHash
  0x042a5fc8c1308c0560a155b11e52c388a8fe6ae419b691949f2a77e440e42fb2
  orderSignature
  0x14ad90564bbff7d9cae1463d0e78d9b1b7d4e3ee134ed835ceb815ed015d556916b5b4d89ff944b42ce1204e27db2def52d91b25d58bf8d79fe8af6066c477b51b
  positionSignatureHash
  0x2224e76396f74ef4922448d1eab8bfd338aa7a95b93439a63a5a3d7409432496

CALLING -> contract_Collateral.approve(address(contract_tradeManager), 10e18);
  contract_DAI.balanceOf(ALICE) -> 210000000000000000000   Initial Alice's DAI balance = 210e18
  contract_DAI.balanceOf(contract_liquidityPool0) -> 20059880239520958083
  contract_DAI.balanceOf(contract_liquidityPool1) -> 20059880239520958083

CALLING -> contract_tradeManager.openPositionViaSignature(_openPosOrder, updateData, ALICE, orderSignature);
  contract_DAI.balanceOf(ALICE) -> 204000000000000000000
  contract_DAI.balanceOf(contract_liquidityPool0) -> 20073516603157321703
  contract_DAI.balanceOf(contract_liquidityPool1) -> 20073516603157321703
  ---------------------------------------------------------------------------------------------------------------------------
  Position details
     id -> 1
     margin -> 4545454545454545455
     volume -> 45454545454545455500
     assetAmount -> 450045004500450045049
     leverage -> 100000000
     isShort -> false
     entryPrice -> 1010000000000000000
     liquidationPrice -> 1010000000000000000
     totalFeeAmount -> 0
  contract_tradeManager.positionIsLiquidatable(contract_tradePair0, 1) -> true
  ---------------------------------------------------------------------------------------------------------------------------
  contract_DAI.balanceOf(ALICE) before liquidation -> 204000000000000000000
  contract_DAI.balanceOf(contract_liquidityPool0) -> 20073516603157321703
  contract_DAI.balanceOf(contract_liquidityPool1) -> 20073516603157321703

CALLING -> contract_tradeManager.liquidatePosition(address(contract_tradePair0), 1, updateData);
  contract_DAI.balanceOf(ALICE) after liquidation -> 213999999999999999999   Final Alice's DAI balance = 214e18
  contract_DAI.balanceOf(contract_liquidityPool0) -> 19800789330430048993 1
  contract_DAI.balanceOf(contract_liquidityPool1) -> 19800789330430048993 1
```

Risk Level:

Likelihood - 2

Impact - 5

Recommendation:

It is recommended to add the `_verifyPositionsValidity(positionId_);` check in the `TradePair._openPosition()` function.

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

During the second review, it has been observed that this change was committed on PR #121 and merged into the master branch.

3.3 (HAL-03) USE OF DEPRECATED CHAINLINK FUNCTION: LATESTANSWER - MEDIUM

Description:

According to Chainlink's documentation (API Reference), the latestAnswer function is deprecated. This function does not throw an error if no answer has been reached, but instead returns 0, possibly causing an incorrect price to be fed to the different price feeds or even a Denial of Service by a division by zero.

Code Location:

ChainlinkUsdPriceFeed.sol

Listing 7: ChainlinkUsdPriceFeed.sol (Line 24)

```
23 function price() external view returns (int256) {
24     return chainlinkPriceFeed.latestAnswer();
25 }
```

UnlimitedPriceFeed.sol

Listing 8: UnlimitedPriceFeed.sol (Line 117)

```
90 function update(bytes calldata updateData_) external {
91     require(updateData_.length == DATA_LENGTH, "UnlimitedPriceFeed
↳ ::update: Bad data length");
92
93     PriceData memory newPriceData = abi.decode(updateData_[
↳ SIGNER_END:], (PriceData));
94
95     // Verify new price data is more recent than the current price
↳  data
96     if (newPriceData.createdOn <= priceData.createdOn) {
97         return;
98     }
```

```
 99
100     // verify signer access controlls
101     address signer = abi.decode(updateData_[SIGNATURE_END:
 ↳ SIGNER_END], (address));
102
103     // throw if the signer is not allowed to update the price
104     _verifySigner(signer);
105
106     // verify signature
107     bytes calldata signature = updateData_[:SIGNATURE_END];
108     require(
109         SignatureChecker.isValidSignatureNow(signer,
 ↳ _hashPriceDataUpdate(newPriceData), signature),
110         "UnlimitedPriceFeed::update: Bad signature"
111     );
112
113     // verify validity of data
114     _verifyValidTo(newPriceData.validTo);
115
116     // verify price deviation is not too high
117     int256 chainlinkPrice = chainlinkPriceFeed.latestAnswer();
118
119     unchecked {
120         int256 maxAbsoluteDeviation = int256(uint256(
 ↳ chainlinkPrice) * maxDeviation / FULL_PERCENT);
121
122         require(
123             newPriceData.price >= chainlinkPrice -
 ↳ maxAbsoluteDeviation
124                 && newPriceData.price <= chainlinkPrice +
 ↳ maxAbsoluteDeviation,
125             "UnlimitedPriceFeed::update: Price deviation too high"
126         );
127     }
128
129     priceData = newPriceData;
130 }
```

UnlimitedPriceFeedAdapter.sol

In this contract a Denial of Service would occur if collateralChainlinkPriceFeed.latestAnswer() returned zero as a division by zero would occur in the _getChainlinkPrice() internal function. This would DoS any call to the UnlimitedPriceFeedAdapter.update() function.

Listing 9: UnlimitedPriceFeedAdapter.sol (Line 144)

```
143 function _assetToUsd(uint256 assetAmount_) private view returns (
 ↳ uint256) {
144     return assetAmount_ * uint256(assetChainlinkPriceFeed.
 ↳ latestAnswer()) / ASSET_MULTIPLIER;
145 }
```

Listing 10: UnlimitedPriceFeedAdapter.sol (Line 170)

```
169 function _collateralToUsd(uint256 collateralAmount_) private view
 ↳ returns (uint256) {
170     return collateralAmount_ * uint256(
 ↳ collateralChainlinkPriceFeed.latestAnswer()) /
 ↳ COLLATERAL_MULTIPLIER;
171 }
```

Listing 11: UnlimitedPriceFeedAdapter.sol (Lines 222,223)

```
221 function _getChainlinkPrice() internal view returns (int256) {
222     return assetChainlinkPriceFeed.latestAnswer() * int256(
 ↳ _priceMultiplier)
223         / collateralChainlinkPriceFeed.latestAnswer();
224 }
```

Risk Level:

Likelihood - 2
Impact - 5

Recommendation:

It is recommended to use Chainlink's latestRoundData() function to get the price instead. It is also recommended to add checks on the return data with proper revert messages if the price is stale or the round is incomplete, for example:

Listing 12: latestRoundData example call

```
(uint80 baseRoundID, int256 answer, , uint256 baseTimestamp,
↳ uint80 baseAnsweredInRound) = assetChainlinkPriceFeed.latestAnswer
↳ ();
require(answer > 0, "ChainlinkPriceOracle: answer <= 0");
require(baseAnsweredInRound >= baseRoundID, "ChainlinkPriceOracle:
↳  Stale price");
require(baseTimestamp > 0, "ChainlinkPriceOracle: Round not
↳ complete");
uint256 _price = uint256(answer);
```

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

During the second review, it has been observed that this change was committed on PR #121 and merged into the master branch.

3.4 (HAL-04) UNLIMITEDPRICEFEED IS VULNERABLE TO CROSSCHAIN SIGNATURE REPLAY ATTACKS - MEDIUM

Description:

The UnlimitedPriceFeedUpdater smart contract contains the function update() which is used to update the price of the collateral with a valid signature:

Listing 13: UnlimitedPriceFeedUpdater.sol (Lines 90-93)

```
67 /**
68  * @notice Update price with signed data.
69  * @param updateData_ Data bytes consisting of signature, signer,
↳ and price data in respected order.
70  */
71 function update(bytes calldata updateData_) external {
72     require(updateData_.length == DATA_LENGTH, "
↳ UnlimitedPriceFeedUpdater::update: Bad data length");
73
74     PriceData memory newPriceData = abi.decode(updateData_[
↳ SIGNER_END:], (PriceData));
75
76     // Verify new price data is more recent than the current price
↳  data
77     // Return if the new price data is not more recent
78     if (newPriceData.createdOn <= priceData.createdOn) {
79         return;
80     }
81
82     // verify signer access controlls
83     address signer = abi.decode(updateData_[SIGNATURE_END:
↳ SIGNER_END], (address));
84
85     // throw if the signer is not allowed to update the price
86     _verifySigner(signer);
87
88     // verify signature
89     bytes calldata signature = updateData_[:SIGNATURE_END];
90     require(
```

```
91          SignatureChecker.isValidSignatureNow(signer,
↳ _hashPriceDataUpdate(newPriceData), signature),
92          "UnlimitedPriceFeedUpdater::update: Bad signature"
93      );
94
95      // verify validity of data
96      _verifyValidTo(newPriceData.validTo);
97
98      _verifyNewPrice(newPriceData.price);
99
100     priceData = newPriceData;
101 }
```

The signature hash is built through the _hashPriceDataUpdate() function:

Listing 14: UnlimitedPriceFeedUpdater.sol (Line 106)

```
105 function _hashPriceDataUpdate(PriceData memory priceData_)
 ↳ internal view returns (bytes32) {
106     return keccak256(abi.encode(address(this), priceData_));
107 }
```

The PriceData struct is also built as shown below:

Listing 15: UnlimitedPriceFeedUpdater.sol

```
 9 /**
10  * @notice Struct to store the price feed data.
11  * @custom:member createdOn The timestamp when the price data was
 ↳ stored.
12  * @custom:member validTo The timestamp until which the price data
 ↳  is valid.
13  * @custom:member price The price.
14  */
15 struct PriceData {
16     uint32 createdOn;
17     uint32 validTo;
18     int192 price;
19 }
```

This implementation:

1. Mitigates any risk of reusing an old signature, as the update() function verifies that the new price data is more recent than the current price data.
2. Verifies that the data is valid through the _verifyValidTo() function.
3. Verifies that the price does not deviate too much by comparing the new price with the price of a Chainlink oracle.

Although, this implementation does not use any type of Domain Separator, making it vulnerable to cross-chain replay attacks. This would be an issue under the following conditions:
1. Unlimited Leverage is deployed in multiple blockchains (i.e. Arbitrum & Polygon), under the same contract addresses.
2. The same signer is set in the Controller for both blockchains.
3. A malicious user would be able to use signatures crafted for Arbitrum in the Polygon contract and vice versa.

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended to use a Domain Separator when creating the PriceData signature.

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

During the second review, it has been observed that this change was committed on PR #121 and merged into the master branch.

3.5 (HAL-05) CENTRALIZATION RISK: ORDEREXECUTOR COULD CLOSE POSITIONS IN HIS BENEFIT BEFORE EVERY PRICE UPDATE - MEDIUM

Description:

The Unlimited Leverage protocol will use the UnlimitedPriceFeedAdapter as a price feed. This price feed will be updated by an authorized signer through the UnlimitedPriceFeedUpdater.update() function:

Listing 16: UnlimitedPriceFeedUpdater.sol (Lines 90-93)

```
67 /**
68  * @notice Update price with signed data.
69  * @param updateData_ Data bytes consisting of signature, signer
↳ and price data in respected order.
70  */
71 function update(bytes calldata updateData_) external {
72     require(updateData_.length == DATA_LENGTH, "
↳ UnlimitedPriceFeedUpdater::update: Bad data length");
73
74     PriceData memory newPriceData = abi.decode(updateData_[
↳ SIGNER_END:], (PriceData));
75
76     // Verify new price data is more recent than the current price
↳  data
77     // Return if the new price data is not more recent
78     if (newPriceData.createdOn <= priceData.createdOn) {
79         return;
80     }
81
82     // verify signer access controlls
83     address signer = abi.decode(updateData_[SIGNATURE_END:
↳ SIGNER_END], (address));
84
85     // throw if the signer is not allowed to update the price
86     _verifySigner(signer);
87
88     // verify signature
```

```
89      bytes calldata signature = updateData_[:SIGNATURE_END];
90      require(
91          SignatureChecker.isValidSignatureNow(signer,
↳ _hashPriceDataUpdate(newPriceData), signature),
92          "UnlimitedPriceFeedUpdater::update: Bad signature"
93      );
94
95      // verify validity of data
96      _verifyValidTo(newPriceData.validTo);
97
98      _verifyNewPrice(newPriceData.price);
99
100     priceData = newPriceData;
101 }
```

On the other hand, all the different orders will be performed by an account with the OrderExecutor role in the TradeManagerOrders contract. For example:

Listing 17: TradeManagerOrders.sol (Lines 40,41)

```
29 /**
30  * @notice Opens a position with a signature
31  * @param order_ Order struct
32  * @param maker_ address of the maker
33  * @param signature_ signature of order_ by maker_
34  */
35 function openPositionViaSignature(
36     OpenPositionOrder calldata order_,
37     UpdateData[] calldata updateData_,
38     address maker_,
39     bytes calldata signature_
40 ) external onlyOrderExecutor onlyActiveTradePair(order_.params.
↳ tradePair) returns (uint256) {
41     _updateContracts(updateData_);
42     _processSignature(order_, maker_, signature_);
43     _verifyConstraints(
44         order_.params.tradePair, order_.constraints, order_.params
↳ .isShort ? UsePrice.MAX : UsePrice.MIN
45     );
46
47     _transferOrderReward(order_.params.tradePair, maker_, msg.
↳ sender);
```

```
48
49      uint256 positionId = _openPosition(order_.params, maker_);
50
51      sigHashToTradeId[keccak256(signature_)] = TradeId(order_.
↳ params.tradePair, uint96(positionId));
52
53      emit OpenedPositionViaSignature(order_.params.tradePair,
↳ positionId, signature_);
54
55      return positionId;
56  }
```

As seen, `_updateContracts(updateData_);` is called:

Listing 18: TradePair.sol (Line 367)

```
360 function _updateContracts(UpdateData[] calldata updateData_)
↳ internal {
361     for (uint256 i; i < updateData_.length; i++) {
362         require(
363             controller.isUpdatable(updateData_[i].
↳ updatableContract),
364             "TradeManager::_updateContracts: Contract not
↳ updatable"
365         );
366
367         IUpdatable(updateData_[i].updatableContract).update(
↳ updateData_[i].data);
368     }
369 }
```

If this is given a valid signature, the UnlimitedPriceFeed will be updated with a new price and then the orders will be processed.

Based on this implementation, Unlimited Leverage could be abused by the OrderExecutor account in the following way:
1. OrderExecutor opens multiple long/short positions.
2. Every time the OrderExecutor gets some UpdateData, the OrderExecutor checks the new price:
- If the new price is higher, OrderExecutor closes all his short positions by calling closePositionViaSignature() with an empty updateData_ and then

submits a new transaction with the updateData_ that updates the price accordingly.
- If the new price is lower, OrderExecutor closes all his long positions by calling closePositionViaSignature() with an empty updateData_ and then submits a new transaction with the updateData_ that updates the price accordingly.

This way, the OrderExecutor is front-running every price update, and he will always profit, reducing the income of the LPs in his benefit.

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

There is no valid recommendation for this issue, as it would require a total re-design of the project.

Remediation Plan:

RISK ACCEPTED: This issue has not been addressed in the latest Commit ID: ae36e3ddea25900d66245c82f88aaafc92266c05

3.6 (HAL-06) FIRST LIQUIDITYPOOL DEPOSITOR COULD BE FRONT-RUN - LOW

Description:

The LiquidityPool contract follows the EIP4626 standard: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/extensions/ERC4626.sol

This extension allows the minting and burning of "shares" (represented using the ERC20 inheritance) in exchange for underlying "assets" through standardized deposit, mint, redeem and burn workflows. But this extension also has the following problem:

When the vault is empty or nearly empty, deposits are at high risk of being stolen through front-running with a "donation" to the vault that inflates the price of a share. This is variously known as a donation or inflation attack and is essentially a problem of slippage. Vault deployers can protect against this attack by making an initial deposit of a non-trivial amount of the asset, such that price manipulation becomes infeasible. Withdrawals may similarly be affected by slippage.

In order to prevent this, the LiquidityPool contract has 2 different mechanisms:

- In the deposit() function, a minShares_ parameter is used. If the final shares that will be given to the user are less than minShares_ the call would revert. As long as the users use correctly the minShares_ parameter, they will be protected against a first deposit front-run.

Listing 19: LiquidityPool.sol (Line 236)

```
229 /**
230  * @notice Deposits an amount of the collateral asset.
231  * @param assets_ The amount of the collateral asset to deposit.
232  * @param minShares_ The desired minimum amount to receive in
```

```
 ↳ exchange for the deposited collateral. Reverts otherwise.
233  * @return The amount of shares received for the deposited
 ↳ collateral.
234  */
235 function deposit(uint256 assets_, uint256 minShares_) external
 ↳ updateUser(msg.sender) returns (uint256) {
236     return _depositAsset(assets_, minShares_, msg.sender);
237 }
```

Listing 20: LiquidityPool.sol (Line 276)

```
270 /**
271  * @dev deposits assets into the pool
272  */
273 function _depositAsset(uint256 assets_, uint256 minShares_,
 ↳ address receiver_) private returns (uint256) {
274     uint256 shares = previewDeposit(assets_);
275
276     require(shares >= minShares_, "LiquidityPool::_depositAsset:
 ↳ Bad slippage");
277
278     _deposit(msg.sender, receiver_, assets_, shares);
279
280     return shares;
281 }
```

- On the other hand, the LiquidityPoolVault uses a 24 decimal precision for the shares minted/burnt, while the most ERC20 tokens use an 18 decimals precision. This means that the initial deposit will be given 1000000 shares per asset (in case that the asset has 18 decimals): (asset * 10e24)/ 10e18

Listing 21: LiquidityPoolVault.sol (Line 23)

```
23 uint8 internal constant _decimals = 24;
```

Listing 22: LiquidityPoolVault.sol (Lines 103,104)

```
95 /**
96  * @dev Internal convertion function (from assets to shares) with
```

```
↳ support for rounding direction
 *
 * Will revert if assets > 0, totalSupply > 0 and totalAssets = 0.
↳  That corresponds to a case where any asset
 * would represent an infinite amout of shares.
 */
function _convertToShares(uint256 assets, Math.Rounding rounding)
↳ internal view virtual returns (uint256 shares) {
    uint256 supply = totalSupply();
    return (assets == 0 || supply == 0)
        ? assets.mulDiv(10 ** decimals(), 10 ** _asset.decimals(),
↳ rounding)
        : assets.mulDiv(supply, totalAssets(), rounding);
}
```

Although, there are some tokens that have more than 18 decimals. For example, YamV2.

The LiquidityPool contract does not prevent anywhere in the code that the token used as the collateral contains more than 18 decimals:

Listing 23: LiquidityPoolVault.sol (Lines 28-30)

```
abstract contract LiquidityPoolVault is ERC20Upgradeable,
↳ ILiquidityPoolVault {
    using Math for uint256;

    IERC20Metadata internal immutable _asset;

    uint8 internal constant _decimals = 24;

    /**
     * @dev Set the underlying asset contract. This must be an
↳ ERC20-compatible contract (ERC20 or ERC777).
     */
    constructor(IERC20Metadata asset_) {
        _asset = asset_;
    }
```

Proof of Concept:

In the case that a token with for example 24 decimals is used and that the initial depositor does not properly make use of the deposit().minShares_ parameter, this inflation attack would be possible:

```
[PASS] test_depositorFrontrun() (gas: 548449)
Logs:
  Initial user balances
  contract_YAMv2.balanceOf(user1) -> 2000000000000000000000000000
  contract_YAMv2.balanceOf(user2) -> 2000000000000000000000000000
  ──────────────────────────────────────────────────────────────────────────────
  Attacker deposits 1 Wei of YamV2
            contract_LiquidityPool.totalSupply() -> 1
            contract_YAMv2.balanceOf(address(contract_LiquidityPool)) -> 1
  ──────────────────────────────────────────────────────────────────────────────
  Attacker transfers 100 YamV2 to the contract
            contract_YAMv2.balanceOf(address(contract_LiquidityPool)) -> 100000000000000000000000001
  ──────────────────────────────────────────────────────────────────────────────
  Victim deposits 200 YamV2
            contract_LiquidityPool.totalSupply() -> 2
            contract_YAMv2.balanceOf(address(contract_LiquidityPool)) -> 300000000000000000000000001
            _user1PoolDetails.totalPoolShares -> 1000000000000
            _user1PoolDetails.unlockedPoolShares -> 1000000000000
            _user1PoolDetails.totalShares -> 1
            _user1PoolDetails.unlockedShares -> 1
            _user1PoolDetails.totalAssets -> 150000000000000000000000000
            _user1PoolDetails.totalAssets -> 150000000000000000000000000
            contract_YAMv2.balanceOf(user1) -> 999999999999999999999999
  ──────────────────────────────────────────────────────────────────────────────
  Attacker withdraws 1 share
            _user1PoolDetails.totalPoolShares -> 999999999999
            _user1PoolDetails.unlockedPoolShares -> 999999999999
            _user1PoolDetails.totalShares -> 0
            _user1PoolDetails.unlockedShares -> 0
            _user1PoolDetails.totalAssets -> 0
            _user1PoolDetails.totalAssets -> 0
            contract_YAMv2.balanceOf(user1) -> 999999999999999999999999
  ──────────────────────────────────────────────────────────────────────────────
            _user2PoolDetails.totalPoolShares -> 1000000000000
            _user2PoolDetails.unlockedPoolShares -> 1000000000000
            _user2PoolDetails.totalShares -> 1
            _user2PoolDetails.unlockedShares -> 1
            _user2PoolDetails.totalAssets -> 150000000000000000000000000
            _user2PoolDetails.totalAssets -> 150000000000000000000000000
            contract_YAMv2.balanceOf(user1) -> 999999999999999999999999
            contract_YAMv2.balanceOf(user2) -> 0
  Victim withdraws all his shares
            _user2PoolDetails.totalPoolShares -> 0
            _user2PoolDetails.unlockedPoolShares -> 0
            _user2PoolDetails.totalShares -> 0
            _user2PoolDetails.unlockedShares -> 0
            _user2PoolDetails.totalAssets -> 0
            _user2PoolDetails.totalAssets -> 0
            contract_YAMv2.balanceOf(user2) -> 150000000000000000000000000   -50e24 YamV2 tokens
            contract_YAMv2.balanceOf(address(contract_LiquidityPool)) -> 150000000000000000000000001
  ──────────────────────────────────────────────────────────────────────────────
  Attacker withdraws all his shares
            _user1PoolDetails.totalPoolShares -> 0
            _user1PoolDetails.unlockedPoolShares -> 0
            _user1PoolDetails.totalShares -> 0
            _user1PoolDetails.unlockedShares -> 0
            _user1PoolDetails.totalAssets -> 0
            _user1PoolDetails.totalAssets -> 0
            contract_YAMv2.balanceOf(user1) -> 250000000000000000000000000   +50e24 YamV2 tokens
            contract_YAMv2.balanceOf(address(contract_LiquidityPool)) -> 0
  ──────────────────────────────────────────────────────────────────────────────
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

It is recommended to ensure that no token with more than 18 decimals can be used as collateral in the LiquidityPool contract. For example:

Listing 24: LiquidityPoolVault.sol (Line 29)

```
18 abstract contract LiquidityPoolVault is ERC20Upgradeable,
↳ ILiquidityPoolVault {
19     using Math for uint256;
20
21     IERC20Metadata internal immutable _asset;
22
23     uint8 internal constant _decimals = 24;
24
25     /**
26      * @dev Set the underlying asset contract. This must be an
↳ ERC20-compatible contract (ERC20 or ERC777).
27      */
28     constructor(IERC20Metadata asset_) {
29         require(asset_.decimals() <= 18, "Collateral decimals must
↳  be <= 18");
30         _asset = asset_;
31     }
```

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

3.7 (HAL-07) MULTIPLE FUNCTIONS DO NOT CHECK THAT THE NEW MARGIN IS HIGHER THAN THE MINIMUM MARGIN - LOW

Description:

In the TradeManagerOrders contract, the function removeMarginFromPositionViaSignature() is used to remove some margin from an existing position in Unlimited Leverage. This function is missing the verifyLeverage(positionId_); modifier. As this modifier is missing, the new margin may be less than the minimum margin permitted by the protocol.

The same issue is also present in the functions:
- partiallyClosePositionViaSignature()
- extendPositionViaSignature()
- extendPositionToLeverageViaSignature()

Code Location:

Listing 25: TradePair.sol (Line 481)

```
467 /**
468  * @notice Removes margin from a position
469  * @param maker_ owner of the position
470  * @param positionId_ id of the position
471  * @param removedMargin_ the margin to be removed
472  */
473 function removeMarginFromPosition(address maker_, uint256
 ↳ positionId_, uint256 removedMargin_)
474     external
475     onlyTradeManager
476     verifyOwner(maker_, positionId_)
477     syncFeesBefore
478     updatePositionFees(positionId_)
479     onlyValidAlteration(positionId_)
480 {
```

```
481     _removeMarginFromPosition(maker_, positionId_, removedMargin_)
 ↳ ;
482 }
483
484 function _removeMarginFromPosition(address maker_, uint256
 ↳ positionId_, uint256 removedMargin_) private {
485     Position storage position = positions[positionId_];
486
487     // update position in storage
488     position.removeMargin(removedMargin_);
489
490     // update aggregated values
491     positionStats.removeTotalCount(removedMargin_, 0, 0, position.
 ↳ isShort);
492
493     _payoutToMaker(maker_, int256(removedMargin_), positionId_);
494
495     emit AlteredPosition(
496         PositionAlterationType.removeMargin,
497         positionId_,
498         position.lastNetMargin(),
499         position.volume,
500         position.assetAmount
501         );
502 }
```

Proof of Concept:

```
------------------------------------------------------------------------------------------------------------------------
Position details
   id -> 1
   margin -> 99890120867046249126
   volume -> 109879132953750874038
   assetAmount -> 108791220746287994097
   leverage -> 1099999
   isShort -> false
   entryPrice -> 1010000000000000000
   liquidationPrice -> 137777772727272728
   totalFeeAmount -> 0
contract_tradeManager.positionIsLiquidatable(contract_tradePair0, 1) -> false
------------------------------------------------------------------------------------------------------------------------
removeMarginOrderHash
0x93a157fcb41149ca76340652c9497c903ecfbe1501b1c1bb16597ec5d58ea9da
orderPartSignature
0x8ae844fa32b617569f1a2cca800486d4bf47873af6ae0735686dd313d9bfe0720c1f045ca95f441451dcfa8915f9e6a0d61b6f8c42d49e8f022e8df3dd27e5511c
positionRemoveMarginSignatureHash
0x059641e15b0e2b238194fb9473ffa129bb26a2c61838df9863b4bb0086d45705
contract_DAI.balanceOf(ALICE) -> 10700000000000000000
contract_DAI.balanceOf(contract_liquidityPool0) -> 200631766135095706102
contract_DAI.balanceOf(contract_liquidityPool1) -> 200631766135095706102
CALLING -> contract_tradeManager.removeMarginFromPositionViaSignature( removeMarginPosOrder, updateData, ALICE, orderPartSignature)
contract_DAI.balanceOf(ALICE) -> 19596320000000000000
contract_DAI.balanceOf(contract_liquidityPool0) -> 200642806135095706102
contract_DAI.balanceOf(contract_liquidityPool1) -> 200642806135095706102
------------------------------------------------------------------------------------------------------------------------
contract_tradePair0.minMargin() -> 1000000000000000000
Position details
   id -> 1
   margin -> 7890120867046249126
   volume -> 109879132953750874038
   assetAmount -> 108791220746287994097
   leverage -> 13926165
   isShort -> false
   entryPrice -> 1010000000000000000
   liquidationPrice -> 983434245454545455
   totalFeeAmount -> 0
contract_tradeManager.positionIsLiquidatable(contract_tradePair0, 1) -> false
------------------------------------------------------------------------------------------------------------------------
```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

It is recommended to add the `verifyLeverage(positionId_);` modifier to the functions:

- `TradePair.removeMarginFromPosition()`
- `TradePair.partiallyClosePositionViaSignature()`
- `TradePair.extendPositionViaSignature()`
- `TradePair.extendPositionToLeverageViaSignature()`

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

Users are allowed to have a margin under the minimum margin, as long as they do not achieve that by removing margin.

3.8 (HAL-08) ADDING MARGIN TO A POSITION MAY RESULT IN A NEGATIVE LIQUIDATION PRICE - LOW

Description:

When adding margin into a position, it is not checked that the new leverage obtained is higher than the minimum leverage allowed:

Listing 26: TradePair.sol

```
506 /**
507  * @notice Adds margin to a position
508  * @param maker_ owner of the position
509  * @param positionId_ id of the position
510  * @param addedMargin_ the margin to be added
511  */
512 function addMarginToPosition(address maker_, uint256 positionId_,
 ↳ uint256 addedMargin_)
513     external
514     onlyTradeManager
515     verifyOwner(maker_, positionId_)
516     syncFeesBefore
517     updatePositionFees(positionId_)
518     onlyValidAlteration(positionId_)
519 {
520     _addMarginToPosition(maker_, positionId_, addedMargin_);
521 }
522
523 /**
524  * @dev Should have received margin from TradeManager
525  */
526 function _addMarginToPosition(address maker_, uint256 positionId_,
 ↳  uint256 addedMargin_) private {
527     Position storage position = positions[positionId_];
528
529     addedMargin_ = _deductAndTransferOpenFee(maker_, addedMargin_,
 ↳  LEVERAGE_MULTIPLIER, positionId_);
530
531     // change position in storage
532     position.addMargin(addedMargin_);
```

```
533     // update aggregated values
534     positionStats.addTotalCount(addedMargin_, 0, 0, position.
 ↳ isShort);
535
536     emit AlteredPosition(
537         PositionAlterationType.addMargin,
538         positionId_,
539         position.lastNetMargin(),
540         position.volume,
541         position.assetAmount
542         );
543 }
```

This causes that when a leverage lower than the minimum leverage is reached, the liquidation price will be negative:

```
Position details
   id -> 1
   margin -> 99890120867046249126
   volume -> 109879132953750874038
   assetAmount -> 10879122074628799 4097
   leverage -> 1099999
   isShort -> false
   entryPrice -> 101000000000000000
   liquidationPrice -> 137777772727272728
   totalFeeAmount -> 0
contract_tradeManager.positionIsLiquidatable(contract_tradePair0, 1) -> false

addMarginToPosOrderHash
0xaa39734f0f1f89c02b8fc0507b259e551475816f731d74a57877831445ad97b5
addMarginPosOrderSignature
0x97057bf62401d341731959abbbb0ae7f933e787c46e3737dacf016e8a99f5b5959e84c38eb2b306644e99e7981b3b41ed7540b932397679bd17815952dbd319b1b
addMarginPosOrderSignatureHash
0x3ddc819b4a437f4c98877eb450d748f8d9c53aecfb32f18db0ae356f003df6dc
contract_DAI.balanceOf(ALICE) -> 1070000000000000000000
contract_DAI.balanceOf(contract_liquidityPool0) -> 2006317661350957061 02
contract_DAI.balanceOf(contract_liquidityPool1) -> 2006317661350957061 02
contract_DAI.balanceOf(ALICE) -> 400000000000000000
contract_DAI.balanceOf(contract_liquidityPool0) -> 2006617361650657360 72
contract_DAI.balanceOf(contract_liquidityPool1) -> 2006617361650657360 72

Position details
   id -> 1
   margin -> 199790220767146149226
   volume -> 109879132953750874038
   assetAmount -> 10879122074628799 4097
   leverage -> 549972
   isShort -> false
   entryPrice -> 101000000000000000
   liquidationPrice:
-780495771909908273
   totalFeeAmount -> 0
contract_tradeManager.positionIsLiquidatable(contract_tradePair0, 1) -> false
```

This does not have a direct impact on the protocol, as this value is only used in the frontend.

Risk Level:

Likelihood - 4

Impact - 1

Recommendation:

It is recommended to return a liquidation price of 0 in the PositionMaths ._liquidationPrice() function in the case that the liquidation price is negative.

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

During the second review, it has been observed that this change was committed on PR #121 and merged into the master branch.

3.9 (HAL-09) LACK OF A DOUBLE-STEP TRANSFEROWNERSHIP PATTERN - LOW

Description:

The current ownership transfer process for all the contracts inheriting from Ownable or OwnableUpgradeable involves the current owner calling the transferOwnership() function:

Listing 27: Ownable.sol

```
97 function transferOwnership(address newOwner) public virtual
↳ onlyOwner {
98     require(newOwner != address(0), "Ownable: new owner is the
↳ zero address");
99     _setOwner(newOwner);
100 }
```

If the nominated EOA account is not a valid account, it is entirely possible that the owner may accidentally transfer ownership to an uncontrolled account, losing the access to all functions with the onlyOwner modifier.

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

It is recommended to implement a two-step process where the owner nominates an account and the nominated account needs to call an acceptOwnership() function for the transfer of the ownership to fully succeed. This ensures the nominated EOA account is a valid and active account. This can be easily achieved by using OpenZeppelin's Ownable2Step contract instead of Ownable:

Listing 28: Ownable2Step.sol (Lines 52-56)

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.8.0) (access/Ownable2Step.sol)

pragma solidity ^0.8.0;

import "./Ownable.sol";

/**
 * @dev Contract module which provides access control mechanism, where
 * there is an account (an owner) that can be granted exclusive access to
 * specific functions.
 *
 * By default, the owner account will be the one that deploys the contract. This
 * can later be changed with {transferOwnership} and {acceptOwnership}.
 *
 * This module is used through inheritance. It will make available all functions
 * from parent (Ownable).
 */
abstract contract Ownable2Step is Ownable {
    address private _pendingOwner;

    event OwnershipTransferStarted(address indexed previousOwner, address indexed newOwner);

    /**
     * @dev Returns the address of the pending owner.
     */
    function pendingOwner() public view virtual returns (address) {
        return _pendingOwner;
    }

    /**
     * @dev Starts the ownership transfer of the contract to a new account. Replaces the pending transfer if there is one.
     * Can only be called by the current owner.
     */
```

```
35      function transferOwnership(address newOwner) public virtual
↳ override onlyOwner {
36          _pendingOwner = newOwner;
37          emit OwnershipTransferStarted(owner(), newOwner);
38      }
39
40      /**
41       * @dev Transfers ownership of the contract to a new account
↳ (`newOwner`) and deletes any pending owner.
42       * Internal function without access restriction.
43       */
44      function _transferOwnership(address newOwner) internal virtual
↳  override {
45          delete _pendingOwner;
46          super._transferOwnership(newOwner);
47      }
48
49      /**
50       * @dev The new owner accepts the ownership transfer.
51       */
52      function acceptOwnership() external {
53          address sender = _msgSender();
54          require(pendingOwner() == sender, "Ownable2Step: caller is
↳  not the new owner");
55          _transferOwnership(sender);
56      }
57  }
```

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

3.10 (HAL-10) LACK OF DISABLEINITIALIZERS CALL TO PREVENT UNINITIALIZED CONTRACTS - LOW

Description:

Multiple contracts are using the Initializable module from OpenZeppelin. Although, all of them except the UnlimitedOwner protect the initialize() function with the onlyOwner modifier. For this reason and in order to prevent leaving that contract uninitialized OpenZeppelin's documentation recommends adding the _disableInitializers function in the constructor to automatically lock the contracts when they are deployed:

Listing 29: DisableInitializers function

```
/**
 * @dev Locks the contract, preventing any future reinitialization
↳ . This cannot be part of an initializer call.
 * Calling this in the constructor of a contract will prevent that
↳  contract from being initialized or reinitialized
 * to any version. It is recommended to use this to lock
↳ implementation contracts that are designed to be called
 * through proxies.
 */
function _disableInitializers() internal virtual {
    require(!_initializing, "Initializable: contract is
↳ initializing");
    if (_initialized < type(uint8).max) {
        _initialized = type(uint8).max;
        emit Initialized(type(uint8).max);
    }
}
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider calling the _disableInitializers function in the UnlimitedOwner contract's constructor:

Listing 30: Constructor preventing uninitialized contracts

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() {
    _disableInitializers();
}
```

Remediation Plan:

SOLVED: Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

During the second review, it was observed that Add disableInitializers to all initializable contracts change was reverted for all contracts except UnlimitedOwner.

Recent Commits - FeeManager.sol

3.11 (HAL-11) UNLIMITEDPRICEADAPTER LACKS THE ONLYOWNER MODIFIER ON INITIALIZER FUNCTION - LOW

Description:

It was observed that many contracts are upgradeable and that the `initialize()` functions of these upgradeable contracts are wrapped with the `onlyOwner` modifier. However, one contract lacks that modifier.

In this case, an attacker can front-run the transaction initializing the `UnlimitedPriceFeedAdapter` contract transaction and start the contract with the parameters that the attacker will specify.

Code Location:

Listing 31: UnlimitedPriceFeedAdapter.sol (Lines 76,81-86)

```
71 function initialize(
72         string calldata name_,
73         uint256 maxDeviation_,
74         AggregatorV3Interface collateralChainlinkPriceFeed_,
75         AggregatorV3Interface assetChainlinkPriceFeed_
76     ) external initializer {
77         __UnlimitedPriceFeedUpdater_init(name_);
78 
79         name = name_;
80 
81         collateralChainlinkPriceFeed =
↳ collateralChainlinkPriceFeed_;
82         assetChainlinkPriceFeed = assetChainlinkPriceFeed_;
83         assetPriceMultiplier = 10 ** assetChainlinkPriceFeed_.
↳ decimals();
84         collateralPriceMultiplier = 10 **
↳ collateralChainlinkPriceFeed_.decimals();
85 
86         _updateMaxDeviation(maxDeviation_);
87     }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to add `onlyOwner` modifier to `UnlimitedPriceFeedAdapter.initialize()` function to prevent initializing the contract by any other user than deployer.

Remediation Plan:

SOLVED: The `Unlimited Network team` solved this finding by adding the `onlyOwner` modifier to the `initialize()` function.

Commit ID: ae36e3ddea25900d66245c82f88aaafc92266c05

3.12 (HAL-12) UPGRADEABLE CONTRACTS LACK RESERVED SPACE FOR FUTURE UPGRADES - LOW

Description:

All the smart contracts in Unlimited Leverage are upgradeable, but they lack storage gaps.

It is considered a best practice in upgradeable contracts to include a state variable named __gap. This __gap state variable will be used as a reserved space for future upgrades. It allows adding new state variables freely in the future without compromising the storage compatibility with existing deployments.

The size of the __gap array is usually calculated so that the amount of storage used by a contract always adds up to the same number (usually 50 storage slots).

Reference:

OpenZeppelin's storage gap

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to add a state variable named __gap as a reserved space for future upgrades in every upgradeable contract.

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

During the second review, it has been observed that this change was committed on PR #121 and merged into the master branch.

3.13 (HAL-13) POSSIBLE DOS WITH BLOCK GAS LIMIT - LOW

Description:

Every time that a position is closed or liquidated, the following internal function is called:

Listing 32: TradePair.sol (Line 673)

```
667 /**
668  * @dev Deletes position entries from storage.
669  */
670 function _deletePosition(uint256 positionId_) internal {
671     address maker = positions[positionId_].owner;
672     uint256[] storage makerPositions = userToPositionIds[maker];
673     for (uint256 i = 0; i < makerPositions.length; i++) {
674         if (makerPositions[i] == positionId_) {
675             makerPositions[i] = makerPositions[makerPositions.
 ↳ length - 1];
676             makerPositions.pop();
677             break;
678         }
679     }
680     delete positions[positionId_];
681 }
```

This is an antipattern and could be exploited the following way:

1. The user opens 10000 different positions.
2. Position number 9999 is liquidatable.
3. TradePair._liquidatePosition() is called.
4. TradePair._deletePosition(9999) is called.
5. This supposes 9999 iterations over the makerPositions array.
6. Block gas limit is reached.
7. Position cannot be liquidated.

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

It is recommended to remove the `mapping(address => uint256[])public userToPositionIds;` from the `TradePair` contract, as it is only used to display information in the front-end.

Remediation Plan:

SOLVED: The `Unlimited Network team` solved the issue in the following commit ID.

`Commit ID :` `ae36e3ddea25900d66245c82f88aaafc92266c05`.

During the second review, it has been observed that this change was committed on `PR #121` and merged into the `master branch`.

3.14 (HAL-14) INCOMPATIBILITY WITH TRANSFER-ON-FEE OR DEFLATIONARY TOKENS - INFORMATIONAL

Description:

In the LiquidityPool contract, the deposit() and depositAndLock() functions assume that the amount of assets_ is transferred to the smart contract after calling SafeERC20.safeTransferFrom(_asset, caller, address(this), assets);:

Listing 33: LiquidityPool.sol (Line 236)

```
229 /**
230  * @notice Deposits an amount of the collateral asset.
231  * @param assets_ The amount of the collateral asset to deposit.
232  * @param minShares_ The desired minimum amount to receive in
 ↳ exchange for the deposited collateral. Reverts otherwise.
233  * @return The amount of shares received for the deposited
 ↳ collateral.
234  */
235 function deposit(uint256 assets_, uint256 minShares_) external
 ↳ updateUser(msg.sender) returns (uint256) {
236     return _depositAsset(assets_, minShares_, msg.sender);
237 }
```

Listing 34: LiquidityPool.sol (Line 278)

```
273 /**
274  * @dev deposits assets into the pool
275  */
276 function _depositAsset(uint256 assets_, uint256 minShares_,
 ↳ address receiver_) private returns (uint256) {
277     uint256 shares = previewDeposit(assets_);
278
279     require(shares >= minShares_, "LiquidityPool::_depositAsset:
 ↳ Bad slippage");
280
281     _deposit(msg.sender, receiver_, assets_, shares);
```

```
282
283     return shares;
284 }
```

Listing 35: LiquidityPoolVault.sol (Line 129)

```
118 /**
119  * @dev Deposit/mint common workflow
120  */
121 function _deposit(address caller, address receiver, uint256 assets
 ↳ , uint256 shares) internal {
122     // If _asset is ERC777, `transferFrom` can trigger a
 ↳ reenterancy BEFORE the transfer happens through the
123     // `tokensToSend` hook. On the other hand, the `tokenReceived`
 ↳  hook, that is triggered after the transfer,
124     // calls the vault, which is assumed not malicious.
125     //
126     // Conclusion: we need to do the transfer before we mint so
 ↳ that any reentrancy would happen before the
127     // assets are transfered and before the shares are minted,
 ↳ which is a valid state.
128     // slither-disable-next-line reentrancy-no-eth
129     SafeERC20.safeTransferFrom(_asset, caller, address(this),
 ↳ assets);
130     _mint(receiver, shares);
131
132     emit Deposit(caller, receiver, assets, shares);
133 }
```

However, this may not be true if the _asset is a transfer-on-fee token or a deflationary/rebasing token, causing the received amount to be less than the accounted amount leading to a wrong amount of shares minted to the users.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to get the actual received token amount by calculating the difference of the _asset balance before and after the transfer. For example:

Listing 36: LiquidityPoolVault.sol (Lines 129,131,132)

```
118 /**
119  * @dev Deposit/mint common workflow
120  */
121 function _deposit(address caller, address receiver, uint256 assets
 ↳ , uint256 shares) internal {
122     // If _asset is ERC777, `transferFrom` can trigger a
 ↳ reenterancy BEFORE the transfer happens through the
123     // `tokensToSend` hook. On the other hand, the `tokenReceived`
 ↳  hook, that is triggered after the transfer,
124     // calls the vault, which is assumed not malicious.
125     //
126     // Conclusion: we need to do the transfer before we mint so
 ↳ that any reentrancy would happen before the
127     // assets are transfered and before the shares are minted,
 ↳ which is a valid state.
128     // slither-disable-next-line reentrancy-no-eth
129     uint256 beforeBalance = _asset.balanceOf(address(this));
130     SafeERC20.safeTransferFrom(_asset, caller, address(this),
 ↳ assets);
131     uint256 afterBalance = _asset.balanceOf(address(this));
132     require(assets == (afterBalance - beforeBalance), "Transfer-on
 ↳ -fee tokens not allowed");
133     _mint(receiver, shares);
134
135     emit Deposit(caller, receiver, assets, shares);
136 }
```

Remediation Plan:

ACKNOWLEDGED: The Unlimited Network team acknowledged the issue.

3.15 (HAL-15) USE ++I INSTEAD OF I++ IN LOOPS FOR LOWER GAS COSTS - INFORMATIONAL

Description:

In the loops shown below, the variable `i` is incremented using `i++`. It is known that, in loops, using `++i` costs less gas per iteration than `i++`. This also affects variables incremented inside the loop code block.

Code Location:

TradePair.sol
- Line 670:
```
for (uint256 i = 0; i < makerPositions.length; i++){
```

TradePairHelper.sol
- Line 21:
```
for (uint256 i = 0; i < tradePairs_.length; i++){
```
- Line 38:
```
for (uint256 i = 0; i < tradePairs_.length; i++){
```
- Line 41:
```
for (uint256 j = 0; j < positionIds.length; j++){
```
- Line 54:
```
for (uint256 i = 0; i < tradePairs_.length; i++){
```

LiquidityPool.sol
- Line 168:
```
for (uint256 i = 0; i < pools.length; i++){
```
- Line 470:
```
for (newNextIndex = _userPoolInfo.nextIndex; newNextIndex <
depositsCount; newNextIndex++){
```
- Line 520:
```
for (uint256 i; i < multipliedPoolValues.length - 1; i++){
```
- Line 555:

```
for (uint256 i; i < multipliedPoolValues.length; i++){
```

LiquidityPoolAdapter.sol

- Line 84:

```
for (uint256 i; i < liquidityPools_.length; i++){
```

- Line 130:

```
for (uint256 i = 0; i < liquidityPools.length; i++){
```

- Line 160:

```
for (uint256 i = 0; i < poolLiquidities.length; i++){
```

- Line 175:

```
for (uint256 i = 0; i < poolLiquidities.length; i++){
```

- Line 222:

```
for (uint256 i; i < poolLiquidities.length; i++){
```

- Line 230:

```
for (uint256 i; i < poolLiquidities.length; i++){
```

- Line 238:

```
for (uint256 i; i < poolLiquidities.length - 1; i++){
```

UserManager.sol

- Line 137:

```
for (uint256 i = 0; i < feeVolumes_.length; i++){
```

- Line 260:

```
for (uint256 i = 0; i < 30; i++){
```

- Line 355:

```
for (uint256 i = 0; i < feeIndexes.length; i++){
```

- Line 375:

```
for (uint256 i = 0; i < feeIndexes.length; i++){
```

TradeManager.sol

- Line 201:

```
for (uint256 i = 0; i < tradePairs.length; i++){
```

- Line 222:

```
for (uint256 i = 0; i < positionIds.length; i++){
```

- Line 279:

```
for (uint256 i = 0; i < tradePairs_.length; i++){
```

- Line 297:

```
for (uint256 i = 0; i < positionIds_.length; i++){
```

- Line 361:

```
for (uint256 i; i < updateData_.length; i++){
```

PriceFeedAggregator.sol

- Line 98:

```
for (uint256 i = 0; i < priceFeeds.length; i++){
```

Proof of Concept:

For example, based on the following test contract:

Listing 37: Test.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity 0.8.9;
3
4 contract test {
5     function postiincrement(uint256 iterations) public {
6         for (uint256 i = 0; i < iterations; i++) {
7         }
8     }
9     function preiincrement(uint256 iterations) public {
10        for (uint256 i = 0; i < iterations; ++i) {
11        }
12    }
13 }
```

It can be seen the difference in the gas costs:

```
>>> test_contract.postiincrement(1)
Transaction sent: 0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 44
  test.postiincrement confirmed   Block: 13622335   Gas used: 21620 (0.32%)

<Transaction '0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> test_contract.preiincrement(1)
Transaction sent: 0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 45
  test.preiincrement confirmed   Block: 13622336   Gas used: 21593 (0.32%)

<Transaction '0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a'>
>>> test_contract.postiincrement(10)
Transaction sent: 0x98c04430526a59balcf947c114b62666a4417165947d31bf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59balcf947c114b62666a4417165947d31bf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05'>
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use `++i` instead of `i++` to increment the value of an `uint` variable inside a loop. This does not only apply to the iterator variable. It also applies to increments done inside the loop code block.

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

3.16 (HAL-16) UNNEEDED INITIALIZATION OF INTEGER VARIABLES TO 0 - INFORMATIONAL

Description:

As `i` is an `uint256`, it is already initialized to 0. `uint256 i = 0` reassigns the 0 to `i` which wastes gas.

Code Location:

TradePair.sol
- Line 670:
```
for (uint256 i = 0; i < makerPositions.length; i++){
```

TradePairHelper.sol
- Line 21:
```
for (uint256 i = 0; i < tradePairs_.length; i++){
```
- Line 38:
```
for (uint256 i = 0; i < tradePairs_.length; i++){
```
- Line 41:
```
for (uint256 j = 0; j < positionIds.length; j++){
```
- Line 54:
```
for (uint256 i = 0; i < tradePairs_.length; i++){
```

LiquidityPool.sol
- Line 168:
```
for (uint256 i = 0; i < pools.length; i++){
```

LiquidityPoolAdapter.sol
- Line 130:
```
for (uint256 i = 0; i < liquidityPools.length; i++){
```
- Line 160:
```
for (uint256 i = 0; i < poolLiquidities.length; i++){
```
- Line 175:
```
for (uint256 i = 0; i < poolLiquidities.length; i++){
```

UserManager.sol
- Line 137:
`for (uint256 i = 0; i < feeVolumes_.length; i++){`
- Line 260:
`for (uint256 i = 0; i < 30; i++){`
- Line 355:
`for (uint256 i = 0; i < feeIndexes.length; i++){`
- Line 375:
`for (uint256 i = 0; i < feeIndexes.length; i++){`

TradeManager.sol
- Line 201:
`for (uint256 i = 0; i < tradePairs.length; i++){`
- Line 222:
`for (uint256 i = 0; i < positionIds.length; i++){`
- Line 279:
`for (uint256 i = 0; i < tradePairs_.length; i++){`
- Line 297:
`for (uint256 i = 0; i < positionIds_.length; i++){`

PriceFeedAggregator.sol
- Line 98:
`for (uint256 i = 0; i < priceFeeds.length; i++){`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended not to initialize `uint` variables to `0` to reduce the gas costs. For example, use instead:
`for (uint256 i; i < makerPositions.length; ++i){`

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

3.17 (HAL-17) BACKEND ADDRESS WOULD HIGHLY BENEFIT OF TRAILING ZERO BYTES - INFORMATIONAL

Description:

When it comes to addresses in particular, some gas savings can be made if the address has at least four leading zero bytes or eight leading zeroes in hex-encoded format.

For example, using OpenZeppelin's StandardToken as a reference implementation, a standard transfer to an address with no zero bytes costs 51,486 gas. However, a transfer to an address with eight zero bytes only costs 50,974 gas, a difference of 51486 - 50974 = 512 gas. This can also be expressed as 64 * 8.

We have noticed that the address used by the backend in all the tests is 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

As the backend address is the one that will be constantly executing all the transactions related to the update/creation/closure of orders, it is recommended to use an address with at least eight leading zeroes. This type of address can be generated by using a vanity address generation tool like Profanity2. Special attention must be taken to not use a vulnerable vanity address generation tool.

Reference :
Article: Efficient Ethereum address

Remediation Plan:

ACKNOWLEDGED: The Unlimited Network team acknowledged the issue.

3.18 (HAL-18) DETAILSOFPOSITION FUNCTION CAN BE OPTIMIZED - INFORMATIONAL

Description:

The TradePair.detailsOfPosition() function can be optimized as it writes twice the same value in the positionDetails.volume:

Listing 38: TradePair.sol (Lines 899,903)

```
884 /**
885  * @notice returns the details of a position
886  * @dev returns PositionDetails
887  */
888 function detailsOfPosition(uint256 positionId_) external view
 ↳ returns (PositionDetails memory) {
889     Position storage position = positions[positionId_];
890     require(position.exists(), "TradePair::detailsOfPosition:
 ↳ Position does not exist");
891
892     // Fee integrals
893     (int256 currentBorrowFeeIntegral, int256
 ↳ currentFundingFeeIntegral) = _getCurrentFeeIntegrals(position.
 ↳ isShort);
894
895     // Construnct position info
896     PositionDetails memory positionDetails;
897     positionDetails.id = positionId_;
898     positionDetails.margin = position.currentNetMargin(
 ↳ currentBorrowFeeIntegral, currentFundingFeeIntegral);
899     positionDetails.volume = position.volume;
900     positionDetails.assetAmount = position.assetAmount;
901     positionDetails.isShort = position.isShort;
902     positionDetails.leverage = position.currentNetLeverage(
 ↳ currentBorrowFeeIntegral, currentFundingFeeIntegral);
903     positionDetails.volume = position.volume;
904     positionDetails.entryPrice = position.entryPrice();
905     positionDetails.liquidationPrice =
906         position.liquidationPrice(currentBorrowFeeIntegral,
 ↳ currentFundingFeeIntegral, absoluteMaintenanceMargin());
```

```
907     positionDetails.totalFeeAmount =
908         position.currentTotalFeeAmount(currentBorrowFeeIntegral,
 ↳ currentFundingFeeIntegral);
909     return positionDetails;
910 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended simply to remove the redundant assignment to the positionDetails.volume.

Remediation Plan:

SOLVED: The Unlimited Network team solved the issue in the following commit ID.

Commit ID : ae36e3ddea25900d66245c82f88aaafc92266c05.

3.19 (HAL-19) REVERT MESSAGES LONGER THAN 32 BYTES COST EXTRA GAS - INFORMATIONAL

Description:

If message data can fit into 32 bytes, it is always a better idea to use BYTES32 datatype rather than bytes or strings as using that variable is much cheaper in Solidity.

Some revert messages are longer than 32 bytes. In this case, they will cost extra gas.

It is also recommended to use Custom Errors to optimize gas usage:

- Custom Errors - Solidity

Code Location:

All require statements in the protocol.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Check all revert strings and consider using 32 bytes at maximum for revert messages or other strings in the protocol to optimize gas usage instead of long revert messages.

Remediation Plan:

ACKNOWLEDGED: This finding was acknowledged by the Unlimited Network team. It will not be fixed in the current version.

RECOMMENDATIONS OVERVIEW

1. Use mulDiv() from the Uniswap's FullMath library in all the divisions to avoid any precision loss.
2. Add the _verifyPositionsValidity(positionId_); check in the TradePair._openPosition() function.
3. Use Chainlink's latestRoundData() function to get the price in all the Price Feeds.
4. Add checks on the return data of the latestRoundData() with proper revert messages if the price is stale or the round is incomplete.
5. Use a Domain Separator when creating the PriceData signature.
6. Ensure that no token with more than 18 decimals can be used as collateral in the LiquidityPool contract.
7. Add the verifyLeverage(positionId_); modifier to the functions:
    - TradePair.removeMarginFromPosition()
    - TradePair.partiallyClosePositionViaSignature()
    - TradePair.extendPositionViaSignature()
    - TradePair.extendPositionToLeverageViaSignature()
8. Return a liquidation price of 0 in the PositionMaths._liquidationPrice() function in case that the liquidation price is negative.
9. Implement a two-step process for the ownership transfer.
10. Call the _disableInitializers function in the UnlimitedOwner contract's constructor.
11. Add a state variable named __gap as a reserved space for future upgrades in every upgradeable contract.
12. Remove the mapping(address => uint256[])public userToPositionIds; from the TradePair contract.
13. To support transfer-on-fee tokens, get the actual received token amount by calculating the difference of the _asset balance before and after the transfer.
14. Use ++i instead of i++ to increment the value of an uint variable inside a loop.
15. Do not to initialize uint variables to 0 to reduce the gas costs.
16. Use an address with at least eight leading zeroes as the backend address.
17. Remove the redundant assignment to the positionDetails.volume in the TradePair.detailsOfPosition() function.

FUZZ TESTING

FUZZ TESTING

Fuzz testing is a testing technique that involves sending randomly generated or mutated inputs to a target system to identify unexpected behavior or vulnerabilities. In the context of smart contract auditing, fuzz testing can help identify potential security issues by exposing the smart contracts to a wide range of inputs that they may not have been designed to handle.

In this audit, we conducted comprehensive fuzzing tests on the target smart contracts to assess their resilience to unexpected inputs. Our goal was to identify any potential vulnerabilities or flaws that could be exploited by an attacker.

The following section provides a detailed description of the fuzzing methodology we used, the tools we employed, and the findings we uncovered. We believe that this information will be useful in helping the development team to identify and address the identified vulnerabilities, thereby improving the overall security posture of the smart contract.

Foundry is a smart contract development toolchain and is the tool that was used to perform all the fuzz testing.

5.1 FUZZ TESTING SCRIPTS

In order to perform the fuzz testing, 3 different files were created:
- EnvConstants.t.sol: Defines the global constants used in the tests.
- EnvProperties.p.sol: Defines the different properties that were tested.
- EnvTests.t.sol: Actual fuzzing tests.

Also a Mock Price Feed Adapter was used: PriceFeedAdapterMock.sol.

These files were pushed to the following repository:
https://github.com/solidant/unlimited-contracts/tree/harborn-audit-fuzz-testing

Commit ID: 40721b3cdb81a107a3ef84e593a138fcf9319eb6

5.2 SETUP INSTRUCTIONS

1. Copy the EnvConstants.t.sol, EnvProperties.p.sol and EnvTests.t.sol files to the src directory.
2. Copy PriceFeedAdapterMock.sol to src/price-feed/PriceFeedAdapterMock.sol.
3. Add the following function to the TradePair contract:

Listing 39

```
function getPayoutToMaker(uint256 positionId_) public view returns
↳  (uint256) {
    Position storage position_ = positions[positionId_];
    (int256 currentBorrowFeeIntegral, int256
↳ currentFundingFeeIntegral) = _getCurrentFeeIntegrals(position_.
↳ isShort);

    int256 netEquity = position_.currentNetEquity(
        _getCurrentPrice(position_.isShort, true),
↳ currentBorrowFeeIntegral, currentFundingFeeIntegral
    );
    return netEquity > 0 ? uint256(netEquity) : 0;
}
```

4. Modify the foundry.toml file as shown below:

Listing 40

```
[profile.default]
src = 'src'
out = 'out'
libs = ['lib']
optimizer = true
optimizer_runs = 200
test = 'test'
fs_permissions = [{ access = "read-write", path = "./"}]
ffi = true

[fuzz]
max_test_rejects = 65536
```

5. Run forge test -vvvv --match-contract EnvTests.

5.3 RESULTS

PROPERTY: Position margin is always lower than the minimum margin

1. Property violated after partially closing a position.
2. Property violated after extending a position.
3. Property violated after extending a position with leverage.
4. Property violated after removing margin from a position.

PROPERTY: Liquidation price is never lower or equal to 0

1. Property violated after adding margin to a position.

PROPERTY: TotalAssetAmountLimit (limit for the total size of all positions)is respected

OK.

PROPERTY: Position leverage is within min/max range

1. Property violated after adding margin to a position (Margin below minimum margin).

PROPERTY: Position volume is lower than maximum volume
OK.

PROPERTY: The _getPayoutToMaker()is never higher than the balance of the TradePair
1. Precision loss in PositionMaths._partiallyClose() causes the TradePair to not have enough balance to pay the maker when closing the position; hence, the position can never be closed unless new positions are opened.

PROPERTY: User made profit without any price change after opening, xyz, closing a position
OK.

Total issues found:
- HAL01 - PRECISION LOSS IN PARTIALLYCLOSEPOSITION FUNCTION WILL BLOCK THE LAST USER FROM CLOSING THE POSITION
- HAL07 - MULTIPLE FUNCTIONS DO NOT CHECK THAT THE NEW MARGIN IS HIGHER THAN THE MINIMUM MARGIN
- HAL08 - ADDING MARGIN TO A POSITION MAY RESULT IN A NEGATIVE LIQUIDATION PRICE

AUTOMATED TESTING

6.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their ABIS and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

FeeManager.sol

```
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).name (src/interfaces/ITradePair.sol#166) shadows:
        - ITradePair.name() (src/interfaces/ITradePair.sol#89) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).collateral (src/interfaces/ITradePair.sol#167) shadows:
        - ITradePair.collateral() (src/interfaces/ITradePair.sol#91) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).priceFeedAdapter (src/interfaces/ITradePair.sol#169) shadows:
        - ITradePair.priceFeedAdapter() (src/interfaces/ITradePair.sol#97) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).liquidityPoolAdapter (src/interfaces/ITradePair.sol#170) shadows:
        - ITradePair.liquidityPoolAdapter() (src/interfaces/ITradePair.sol#99) (function)
ITradePair.setLiquidatorReward(uint256).liquidatorReward (src/interfaces/ITradePair.sol#179) shadows:
        - ITradePair.liquidatorReward() (src/interfaces/ITradePair.sol#119) (function)
ITradePair.setMinLeverage(uint128).minLeverage (src/interfaces/ITradePair.sol#181) shadows:
        - ITradePair.minLeverage() (src/interfaces/ITradePair.sol#123) (function)
ITradePair.setMaxLeverage(uint128).maxLeverage (src/interfaces/ITradePair.sol#183) shadows:
        - ITradePair.maxLeverage() (src/interfaces/ITradePair.sol#121) (function)
ITradePair.setMinMargin(uint256).minMargin (src/interfaces/ITradePair.sol#185) shadows:
        - ITradePair.minMargin() (src/interfaces/ITradePair.sol#125) (function)
ITradePair.setVolumeLimit(uint256).volumeLimit (src/interfaces/ITradePair.sol#187) shadows:
        - ITradePair.volumeLimit() (src/interfaces/ITradePair.sol#127) (function)
ITradePair.setTotalAssetAmountLimit(uint256).totalAssetAmountLimit (src/interfaces/ITradePair.sol#191) shadows:
        - ITradePair.totalAssetAmountLimit() (src/interfaces/ITradePair.sol#129) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

FeeManager._depositFeesToLiquidityPools(address,IERC20,uint256) (src/fee-manager/FeeManager.sol#393-398) is never used and should be removed
FeeManager._getLiquidityPoolAdapterFromTradePair(address) (src/fee-manager/FeeManager.sol#403-405) is never used and should be removed
FeeManager._onlyValidTradePair() (src/fee-manager/FeeManager.sol#412-414) is never used and should be removed
FeeManager._spreadFees(address,address,IERC20,uint256,address) (src/fee-manager/FeeManager.sol#305-377) is never used and should be removed
UnlimitedOwnable.isOnlimitedOwner() (src/shared/UnlimitedOwnable.sol#35-37) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.17 (src/fee-manager/FeeManager.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (src/interfaces/IController.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IFeeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILiquidityPool.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILiquidityPoolAdapter.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAdapter.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradePair.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUserManager.sol#3) allows old versions
Pragma version0.8.17 (src/shared/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/shared/UnlimitedOwnable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable UnlimitedOwnable.constructor(IUnlimitedOwner)._unlimitedOwner (src/shared/UnlimitedOwnable.sol#20) is too similar to FeeManager.constructor(IUnlimitedOwner,IController,IUserManager).unlimitedOwner_ (src/fee-manager/FeeManager.sol#66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

LiquidityPool.sol

```
LiquidityPool.defaultLockTime (src/liquidity-pools/LiquidityPool.sol#78) is never initialized. It is used in:
        - LiquidityPool.canTransferLps(address) (src/liquidity-pools/LiquidityPool.sol#194-197)
        - LiquidityPool.canWithdrawLps(address) (src/liquidity-pools/LiquidityPool.sol#204-206)
LiquidityPool.earlyWithdrawalFee (src/liquidity-pools/LiquidityPool.sol#81) is never initialized. It is used in:
        - LiquidityPool.userWithdrawalFee(address) (src/liquidity-pools/LiquidityPool.sol#213-215)
LiquidityPool.earlyWithdrawalTime (src/liquidity-pools/LiquidityPool.sol#84) is never initialized. It is used in:
        - LiquidityPool.canTransferLps(address) (src/liquidity-pools/LiquidityPool.sol#194-197)
        - LiquidityPool.userWithdrawalFee(address) (src/liquidity-pools/LiquidityPool.sol#213-215)
LiquidityPool.minimumAmount (src/liquidity-pools/LiquidityPool.sol#87) is never initialized. It is used in:
        - LiquidityPool.availableLiquidity() (src/liquidity-pools/LiquidityPool.sol#149-159)
LiquidityPool.lastDepositTime (src/liquidity-pools/LiquidityPool.sol#93) is never initialized. It is used in:
        - LiquidityPool.canTransferLps(address) (src/liquidity-pools/LiquidityPool.sol#194-197)
        - LiquidityPool.canWithdrawLps(address) (src/liquidity-pools/LiquidityPool.sol#204-206)
        - LiquidityPool.userWithdrawalFee(address) (src/liquidity-pools/LiquidityPool.sol#213-215)
LiquidityPool.userPoolInfo (src/liquidity-pools/LiquidityPool.sol#96) is never initialized. It is used in:
        - LiquidityPool.previewPoolOf(address,uint256) (src/liquidity-pools/LiquidityPool.sol#179-187)
        - LiquidityPool._lockShares(uint256,uint256,address) (src/liquidity-pools/LiquidityPool.sol#286-299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

Reentrancy in LiquidityPoolVault._withdraw(address,address,address,uint256,uint256) (src/liquidity-pools/LiquidityPoolVault.sol#138-149):
        External calls:
        - SafeERC20.safeTransfer(_asset,receiver,assets) (src/liquidity-pools/LiquidityPoolVault.sol#146)
        Event emitted after the call(s):
        - Withdraw(caller,receiver,owner,assets,shares) (src/liquidity-pools/LiquidityPoolVault.sol#148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

LiquidityPool.canTransferLps(address) (src/liquidity-pools/LiquidityPool.sol#194-197) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp - lastDepositTime[user_] >= transferLockTime (src/liquidity-pools/LiquidityPool.sol#196)
LiquidityPool.canWithdrawLps(address) (src/liquidity-pools/LiquidityPool.sol#204-206) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp - lastDepositTime[user_] >= defaultLockTime (src/liquidity-pools/LiquidityPool.sol#205)
LiquidityPool.userWithdrawalFee(address) (src/liquidity-pools/LiquidityPool.sol#213-215) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp - lastDepositTime[user_] < earlyWithdrawalTime (src/liquidity-pools/LiquidityPool.sol#214)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
LiquidityPool._beforeTokenTransfer(address,address,uint256) (src/liquidity-pools/LiquidityPool.sol#568-573) is never used and should be removed
LiquidityPool._canTransfer(address) (src/liquidity-pools/LiquidityPool.sol#702-704) is never used and should be removed
LiquidityPool._getPoolMultipliers() (src/liquidity-pools/LiquidityPool.sol#548-562) is never used and should be removed
LiquidityPool._onlyValidLiquidityPoolAdapter() (src/liquidity-pools/LiquidityPool.sol#710-715) is never used and should be removed
LiquidityPool._previewPoolShareUnlock(address,uint256) (src/liquidity-pools/LiquidityPool.sol#461-478) is never used and should be removed
LiquidityPool._verifyPoolParameters(uint256,uint256) (src/liquidity-pools/LiquidityPool.sol#721-724) is never used and should be removed
LiquidityPoolVault._convertToAssets(uint256,Math.Rounding) (src/liquidity-pools/LiquidityPoolVault.sol#111-116) is never used and should be removed
LiquidityPoolVault._convertToShares(uint256,Math.Rounding) (src/liquidity-pools/LiquidityPoolVault.sol#101-106) is never used and should be removed
LiquidityPoolVault._withdraw(address,address,address,uint256,uint256) (src/liquidity-pools/LiquidityPoolVault.sol#138-149) is never used and should be removed
UnlimitedOwnable.isUnlimitedOwner() (src/shared/UnlimitedOwnable.sol#35-37) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (src/interfaces/IController.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILiquidityPool.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILiquidityPoolVault.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3) allows old versions
Pragma version0.8.17 (src/liquidity-pools/LiquidityPool.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (src/liquidity-pools/LiquidityPoolVault.sol#3) allows old versions
Pragma version0.8.17 (src/shared/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/shared/UnlimitedOwnable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable LiquidityPoolVault._asset (src/liquidity-pools/LiquidityPoolVault.sol#21) is not in mixedCase
Constant LiquidityPoolVault._decimals (src/liquidity-pools/LiquidityPoolVault.sol#23) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable UnlimitedOwnable.constructor(IUnlimitedOwner)._unlimitedOwner (src/shared/UnlimitedOwnable.sol#20) is too similar to LiquidityPool.constructor(IUnlimitedOwner,IERC20Metadata,IController).unlimitedOwner_ (src/liquidity-pools/Liqu
idityPool.sol#107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

LiquidityPoolVault._decimals (src/liquidity-pools/LiquidityPoolVault.sol#23) is never used in LiquidityPool (src/liquidity-pools/LiquidityPool.sol#62-762)
LiquidityPool.MAXIMUM_MULTIPLIER (src/liquidity-pools/LiquidityPool.sol#68) is never used in LiquidityPool (src/liquidity-pools/LiquidityPool.sol#62-762)
LiquidityPool.MAXIMUM_LOCK_TIME (src/liquidity-pools/LiquidityPool.sol#70) is never used in LiquidityPool (src/liquidity-pools/LiquidityPool.sol#62-762)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

LiquidityPool.defaultLockTime (src/liquidity-pools/LiquidityPool.sol#78) should be constant
LiquidityPool.earlyWithdrawalFee (src/liquidity-pools/LiquidityPool.sol#81) should be constant
LiquidityPool.earlyWithdrawalTime (src/liquidity-pools/LiquidityPool.sol#84) should be constant
LiquidityPool.minimumAmount (src/liquidity-pools/LiquidityPool.sol#87) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

initialize(string,string,uint256,uint256,uint256,uint256) should be declared external:
        - LiquidityPool.initialize(string,string,uint256,uint256,uint256,uint256) (src/liquidity-pools/LiquidityPool.sol#125-139)
availableLiquidity() should be declared external:
        - LiquidityPool.availableLiquidity() (src/liquidity-pools/LiquidityPool.sol#149-159)
canTransferLps(address) should be declared external:
        - LiquidityPool.canTransferLps(address) (src/liquidity-pools/LiquidityPool.sol#194-197)
canWithdrawLps(address) should be declared external:
        - LiquidityPool.canWithdrawLps(address) (src/liquidity-pools/LiquidityPool.sol#204-206)
userWithdrawalFee(address) should be declared external:
        - LiquidityPool.userWithdrawalFee(address) (src/liquidity-pools/LiquidityPool.sol#213-215)
asset() should be declared external:
        - LiquidityPoolVault.asset() (src/liquidity-pools/LiquidityPoolVault.sol#35-37)
convertToShares(uint256) should be declared external:
        - LiquidityPoolVault.convertToShares(uint256) (src/liquidity-pools/LiquidityPoolVault.sol#56-58)
convertToAssets(uint256) should be declared external:
        - LiquidityPoolVault.convertToAssets(uint256) (src/liquidity-pools/LiquidityPoolVault.sol#63-65)
previewMint(uint256) should be declared external:
        - LiquidityPoolVault.previewMint(uint256) (src/liquidity-pools/LiquidityPoolVault.sol#77-79)
previewWithdraw(uint256) should be declared external:
        - LiquidityPoolVault.previewWithdraw(uint256) (src/liquidity-pools/LiquidityPoolVault.sol#84-86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

LiquidityPoolAdapter.sol

```
LiquidityPoolAdapter._updateLiquidityPools(LiquidityPoolConfig[]).i (src/liquidity-pools/LiquidityPoolAdapter.sol#84) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

LiquidityPoolAdapter.constructor(IUnlimitedOwner,IController,address,IERC20).feeManager_ (src/liquidity-pools/LiquidityPoolAdapter.sol#41) lacks a zero-check on :
        - feeManager = feeManager_ (src/liquidity-pools/LiquidityPoolAdapter.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

LiquidityPoolAdapter._depositToLiquidityPool(address,uint256,bool) (src/liquidity-pools/LiquidityPoolAdapter.sol#253-266) is never used and should be removed
LiquidityPoolAdapter._depositToLiquidityPools(uint256,bool) (src/liquidity-pools/LiquidityPoolAdapter.sol#216-251) is never used and should be removed
LiquidityPoolAdapter._onlyFeeManager() (src/liquidity-pools/LiquidityPoolAdapter.sol#276-278) is never used and should be removed
LiquidityPoolAdapter._onlyValidTradePair() (src/liquidity-pools/LiquidityPoolAdapter.sol#270-274) is never used and should be removed
UnlimitedOwnable.isUnlimitedOwner() (src/shared/UnlimitedOwnable.sol#35-37) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (src/interfaces/IController.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILiquidityPool.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILiquidityPoolAdapter.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3) allows old versions
Pragma version0.8.17 (src/liquidity-pools/LiquidityPoolAdapter.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/shared/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/shared/UnlimitedOwnable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable UnlimitedOwnable.constructor(IUnlimitedOwner)._unlimitedOwner (src/shared/UnlimitedOwnable.sol#20) is too similar to LiquidityPoolAdapter.constructor(IUnlimitedOwner,IController,address,IERC20).unlimitedOwner_ (src/liquidity-poo
ls/LiquidityPoolAdapter.sol#41)
Variable LiquidityPoolAdapter._depositToLiquidityPool(address,uint256,bool).liquidityPool_ (src/liquidity-pools/LiquidityPoolAdapter.sol#253) is too similar to LiquidityPoolAdapter.liquidityPools (src/liquidity-pools/LiquidityPoolAdapter
.sol#32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

LiquidityPoolAdapter.maxPayoutProportion (src/liquidity-pools/LiquidityPoolAdapter.sol#29) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

availableLiquidity() should be declared external:
        - LiquidityPoolAdapter.availableLiquidity() (src/liquidity-pools/LiquidityPoolAdapter.sol#128-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

ChainlinkUsdPriceFeed.sol

```
Different versions of Solidity is used:
        - Version used: ['^0.8.17', '^0.8.0']
        - ^0.8.0 (src/external/interfaces/chainlink/AggregatorInterface.sol#2)
        - ^0.8.0 (src/external/interfaces/chainlink/AggregatorV2V3Interface.sol#2)
        - ^0.8.0 (src/external/interfaces/chainlink/AggregatorV3Interface.sol#2)
        - ^0.8.0 (src/interfaces/IPriceFeed.sol#3)
        - 0.8.17 (src/price-feed/ChainlinkUsdPriceFeed.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorInterface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV2V3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version0.8.17 (src/price-feed/ChainlinkUsdPriceFeed.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

ChainlinkUsdPriceFeedPrevious.sol

```
Different versions of Solidity is used:
        - Version used: ['^0.8.17', '^0.8.0']
        - ^0.8.0 (src/external/interfaces/chainlink/AggregatorInterface.sol#2)
        - ^0.8.0 (src/external/interfaces/chainlink/AggregatorV2V3Interface.sol#2)
        - ^0.8.0 (src/external/interfaces/chainlink/AggregatorV3Interface.sol#2)
        - ^0.8.0 (src/interfaces/IPriceFeed.sol#3)
        - 0.8.17 (src/price-feed/ChainlinkUsdPriceFeedPrevious.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorInterface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV2V3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version0.8.17 (src/price-feed/ChainlinkUsdPriceFeedPrevious.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

PriceFeedAdapter.sol

```
PriceFeedAdapter.collateralToAssetMax(uint256) (src/price-feed/PriceFeedAdapter.sol#53-57) performs a multiplication on the result of a division:
        -collateralInDenominator = collateralAmount_ * uint256(collateralPriceFeedAggregator.maxPrice()) / COLLATERAL_MULTIPLIER (src/price-feed/PriceFeedAdapter.sol#54-55)
        -collateralInDenominator * ASSET_MULTIPLIER / uint256(assetPriceFeedAggregator.minPrice()) (src/price-feed/PriceFeedAdapter.sol#56)
PriceFeedAdapter.collateralToAssetMin(uint256) (src/price-feed/PriceFeedAdapter.sol#63-67) performs a multiplication on the result of a division:
        -collateralInDenominator = collateralAmount_ * uint256(collateralPriceFeedAggregator.minPrice()) / COLLATERAL_MULTIPLIER (src/price-feed/PriceFeedAdapter.sol#64-65)
        -collateralInDenominator * ASSET_MULTIPLIER / uint256(assetPriceFeedAggregator.maxPrice()) (src/price-feed/PriceFeedAdapter.sol#66)
PriceFeedAdapter.assetToCollateralMax(uint256) (src/price-feed/PriceFeedAdapter.sol#73-76) performs a multiplication on the result of a division:
        -assetInDenominator = assetAmount_ * uint256(assetPriceFeedAggregator.maxPrice()) / ASSET_MULTIPLIER (src/price-feed/PriceFeedAdapter.sol#74)
        -assetInDenominator * COLLATERAL_MULTIPLIER / uint256(collateralPriceFeedAggregator.minPrice()) (src/price-feed/PriceFeedAdapter.sol#75)
```

```
PriceFeedAdapter.assetToCollateralMin(uint256) (src/price-feed/PriceFeedAdapter.sol#82-85) performs a multiplication on the result of a division:
        -assetInDenominator = assetAmount_ * uint256(assetPriceFeedAggregator.manPrice()) / ASSET_MULTIPLIER (src/price-feed/PriceFeedAdapter.sol#83)
        -assetInDenominator * COLLATERAL_MULTIPLIER / uint256(collateralPriceFeedAggregator.maxPrice()) (src/price-feed/PriceFeedAdapter.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Different versions of Solidity is used:
        - Version used: ['^0.8.17', '^0.8.0']
        - ^0.8.0 (src/interfaces/IPriceFeed.sol#3)
        - ^0.8.0 (src/interfaces/IPriceFeedAdapter.sol#2)
        - ^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3)
        - 0.8.17 (src/price-feed/PriceFeedAdapter.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAdapter.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3) allows old versions
Pragma version0.8.17 (src/price-feed/PriceFeedAdapter.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable PriceFeedAdapter.ASSET_MULTIPLIER (src/price-feed/PriceFeedAdapter.sol#20) is not in mixedCase
Variable PriceFeedAdapter.COLLATERAL_MULTIPLIER (src/price-feed/PriceFeedAdapter.sol#21) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

PriceFeedAggregator.sol

```
Different versions of Solidity is used:
        - Version used: ['^0.8.17', '^0.8.0']
        - ^0.8.0 (src/external/interfaces/chainlink/AggregatorInterface.sol#2)
        - ^0.8.0 (src/external/interfaces/chainlink/AggregatorV2V3Interface.sol#2)
        - ^0.8.0 (src/external/interfaces/chainlink/AggregatorV3Interface.sol#2)
        - ^0.8.0 (src/interfaces/IPriceFeed.sol#3)
        - ^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3)
        - ^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3)
        - 0.8.17 (src/price-feed/PriceFeedAggregator.sol#3)
        - 0.8.17 (src/shared/UnlimitedOwnable.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorInterface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV2V3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3) allows old versions
Pragma version0.8.17 (src/price-feed/PriceFeedAggregator.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/shared/UnlimitedOwnable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable UnlimitedOwnable.constructor(IUnlimitedOwner)._unlimitedOwner (src/shared/UnlimitedOwnable.sol#20) is too similar to PriceFeedAggregator.constructor(IUnlimitedOwner,string,uint256,IPriceFeed[]).unlimitedOwner_ (src/price-feed/Pr
iceFeedAggregator.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

PriceFeedAggregator.USD_MULTIPLIER (src/price-feed/PriceFeedAggregator.sol#17) is never used in PriceFeedAggregator (src/price-feed/PriceFeedAggregator.sol#14-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

UnlimitedPriceFeed.sol

```
UnlimitedPriceFeed.priceData (src/price-feed/UnlimitedPriceFeed.sol#58) is never initialized. It is used in:
        - UnlimitedPriceFeed.price() (src/price-feed/UnlimitedPriceFeed.sol#82-84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

UnlimitedOwnable._onlyOwner() (src/shared/UnlimitedOwnable.sol#40-42) is never used and should be removed
UnlimitedOwnable.isUnlimitedOwner() (src/shared/UnlimitedOwnable.sol#35-37) is never used and should be removed
UnlimitedPriceFeed._hashPriceDataUpdate(PriceData) (src/price-feed/UnlimitedPriceFeed.sol#142-144) is never used and should be removed
UnlimitedPriceFeed._verifySigner(address) (src/price-feed/UnlimitedPriceFeed.sol#161-163) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorInterface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV2V3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IController.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUpdatable.sol#3) allows old versions
Pragma version0.8.17 (src/price-feed/UnlimitedPriceFeed.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/shared/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/shared/UnlimitedOwnable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable UnlimitedOwnable.constructor(IUnlimitedOwner)._unlimitedOwner (src/shared/UnlimitedOwnable.sol#20) is too similar to UnlimitedPriceFeed.constructor(IUnlimitedOwner,AggregatorV2V3Interface,IController,uint256).unlimitedOwner_ (sr
c/price-feed/UnlimitedPriceFeed.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

UnlimitedPriceFeed (src/price-feed/UnlimitedPriceFeed.sol#33-171) does not implement functions:
        - UnlimitedPriceFeed._hashPriceDataUpdate(PriceData) (src/price-feed/UnlimitedPriceFeed.sol#142-144)
        - UnlimitedOwnable._onlyOwner() (src/shared/UnlimitedOwnable.sol#40-42)
        - UnlimitedPriceFeed._updateMaxDeviation(uint256) (src/price-feed/UnlimitedPriceFeed.sol#146-153)
        - UnlimitedPriceFeed._verifySigner(address) (src/price-feed/UnlimitedPriceFeed.sol#161-163)
        - UnlimitedPriceFeed._verifyValidTo(uint256) (src/price-feed/UnlimitedPriceFeed.sol#157-159)
        - UnlimitedOwnable.isUnlimitedOwner() (src/shared/UnlimitedOwnable.sol#35-37)
        - UnlimitedPriceFeed.updateMaxDeviation(uint256) (src/price-feed/UnlimitedPriceFeed.sol#136-138)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

UnlimitedPriceFeed.DATA_LENGTH (src/price-feed/UnlimitedPriceFeed.sol#44) is never used in UnlimitedPriceFeed (src/price-feed/UnlimitedPriceFeed.sol#33-171)
UnlimitedPriceFeed.MINIMUM_MAX_DEVIATION (src/price-feed/UnlimitedPriceFeed.sol#47) is never used in UnlimitedPriceFeed (src/price-feed/UnlimitedPriceFeed.sol#33-171)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

UnlimitedPriceFeed.maxDeviation (src/price-feed/UnlimitedPriceFeed.sol#56) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

price() should be declared external:
        - UnlimitedPriceFeed.price() (src/price-feed/UnlimitedPriceFeed.sol#82-84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

UnlimitedPriceFeedAdapter.sol

```
UnlimitedPriceFeedUpdater.priceData (src/price-feed/UnlimitedPriceFeedUpdater.sol#46) is never initialized. It is used in:
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

Reentrancy in UnlimitedPriceFeedUpdater.update(bytes) (src/price-feed/UnlimitedPriceFeedUpdater.sol#71-101):
        External calls:
        - require(bool,string)(SignatureChecker.isValidSignatureNow(signer,_hashPriceDataUpdate(newPriceData),signature),UnlimitedPriceFeedUpdater::update: Bad signature) (src/price-feed/UnlimitedPriceFeedUpdater.sol#90-93)
        State variables written after the call(s):
        - priceData = newPriceData (src/price-feed/UnlimitedPriceFeedUpdater.sol#100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

UnlimitedPriceFeedUpdater._verifyValidTo(uint256) (src/price-feed/UnlimitedPriceFeedUpdater.sol#113-115) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(validTo_ >= block.timestamp,UnlimitedPriceFeedUpdater::_verifyValidTo: Price is not valid) (src/price-feed/UnlimitedPriceFeedUpdater.sol#114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

UnlimitedOwnable.isUnlimitedOwner() (src/shared/UnlimitedOwnable.sol#35-37) is never used and should be removed
UnlimitedPriceFeedAdapter._getChainlinkPrice() (src/price-feed/UnlimitedPriceFeedAdapter.sol#221-224) is never used and should be removed
UnlimitedPriceFeedAdapter._verifyNewPrice(int256) (src/price-feed/UnlimitedPriceFeedAdapter.sol#208-219) is never used and should be removed
UnlimitedPriceFeedUpdater._hashPriceDataUpdate(PriceData) (src/price-feed/UnlimitedPriceFeedUpdater.sol#105-107) is never used and should be removed
UnlimitedPriceFeedUpdater._verifyNewPrice(int256) (src/price-feed/UnlimitedPriceFeedUpdater.sol#109) is never used and should be removed
UnlimitedPriceFeedUpdater._verifySigner(address) (src/price-feed/UnlimitedPriceFeedUpdater.sol#117-119) is never used and should be removed
UnlimitedPriceFeedUpdater._verifyValidTo(uint256) (src/price-feed/UnlimitedPriceFeedUpdater.sol#113-115) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorInterface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV2V3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/external/interfaces/chainlink/AggregatorV3Interface.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IController.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAdapter.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUpdatable.sol#3) allows old versions
Pragma version0.8.17 (src/price-feed/UnlimitedPriceFeedAdapter.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/price-feed/UnlimitedPriceFeedUpdater.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/shared/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/shared/UnlimitedOwnable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable UnlimitedPriceFeedAdapter.ASSET_MULTIPLIER (src/price-feed/UnlimitedPriceFeedAdapter.sol#34) is not in mixedCase
Variable UnlimitedPriceFeedAdapter.COLLATERAL_MULTIPLIER (src/price-feed/UnlimitedPriceFeedAdapter.sol#35) is not in mixedCase
Variable UnlimitedPriceFeedUpdater._priceMultiplier (src/price-feed/UnlimitedPriceFeedUpdater.sol#44) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable UnlimitedOwnable.constructor(IUnlimitedOwner)._unlimitedOwner (src/shared/UnlimitedOwnable.sol#20) is too similar to UnlimitedPriceFeedAdapter.constructor(string,AggregatorV2V3Interface,AggregatorV2V3Interface,uint256,uint256,ui
nt256,IController,uint256,IUnlimitedOwner).unlimitedOwner_ (src/price-feed/UnlimitedPriceFeedAdapter.sol#58)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
UnlimitedPriceFeedAdapter (src/price-feed/UnlimitedPriceFeedAdapter.sol#22-225) does not implement functions:
        - UnlimitedPriceFeedAdapter._getChainlinkPrice() (src/price-feed/UnlimitedPriceFeedAdapter.sol#221-224)
        - UnlimitedPriceFeedUpdater._hashPriceDataUpdate(PriceData) (src/price-feed/UnlimitedPriceFeedUpdater.sol#105-107)
        - UnlimitedOwnable._onlyOwner() (src/shared/UnlimitedOwnable.sol#40-42)
        - UnlimitedPriceFeedUpdater._price() (src/price-feed/UnlimitedPriceFeedUpdater.sol#63-65)
        - UnlimitedPriceFeedAdapter._verifyNewPrice(int256) (src/price-feed/UnlimitedPriceFeedAdapter.sol#208-219)
        - UnlimitedPriceFeedUpdater._verifySigner(address) (src/price-feed/UnlimitedPriceFeedUpdater.sol#117-119)
        - UnlimitedPriceFeedUpdater._verifyValsTo(uint256) (src/price-feed/UnlimitedPriceFeedUpdater.sol#113-115)
        - UnlimitedOwnable.isUnlimitedOwner() (src/shared/UnlimitedOwnable.sol#35-37)
        - UnlimitedPriceFeedUpdater.update(bytes) (src/price-feed/UnlimitedPriceFeedUpdater.sol#71-101)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

UnlimitedPriceFeedUpdater.DATA_LENGTH (src/price-feed/UnlimitedPriceFeedUpdater.sol#35) is never used in UnlimitedPriceFeedAdapter (src/price-feed/UnlimitedPriceFeedAdapter.sol#22-225)
UnlimitedPriceFeedAdapter.MINIMUM_MAX_DEVIATION (src/price-feed/UnlimitedPriceFeedAdapter.sol#26) is never used in UnlimitedPriceFeedAdapter (src/price-feed/UnlimitedPriceFeedAdapter.sol#22-225)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

UnlimitedPriceFeedAdapter.maxDeviation (src/price-feed/UnlimitedPriceFeedAdapter.sol#37) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

Controller.sol

```
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).name (src/interfaces/ITradePair.sol#166) shadows:
        - ITradePair.name() (src/interfaces/ITradePair.sol#89) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).collateral (src/interfaces/ITradePair.sol#167) shadows:
        - ITradePair.collateral() (src/interfaces/ITradePair.sol#91) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).priceFeedAdapter (src/interfaces/ITradePair.sol#169) shadows:
        - ITradePair.priceFeedAdapter() (src/interfaces/ITradePair.sol#97) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).liquidityPoolAdapter (src/interfaces/ITradePair.sol#170) shadows:
        - ITradePair.liquidityPoolAdapter() (src/interfaces/ITradePair.sol#99) (function)
ITradePair.setLiquidatorReward(uint256).liquidatorReward (src/interfaces/ITradePair.sol#179) shadows:
        - ITradePair.liquidatorReward() (src/interfaces/ITradePair.sol#119) (function)
ITradePair.setMinLeverage(uint128).minLeverage (src/interfaces/ITradePair.sol#181) shadows:
        - ITradePair.minLeverage() (src/interfaces/ITradePair.sol#123) (function)
ITradePair.setMaxLeverage(uint128).maxLeverage (src/interfaces/ITradePair.sol#183) shadows:
        - ITradePair.maxLeverage() (src/interfaces/ITradePair.sol#121) (function)
ITradePair.setMinMargin(uint256).minMargin (src/interfaces/ITradePair.sol#185) shadows:
        - ITradePair.minMargin() (src/interfaces/ITradePair.sol#125) (function)
ITradePair.setVolumeLimit(uint256).volumeLimit (src/interfaces/ITradePair.sol#187) shadows:
        - ITradePair.volumeLimit() (src/interfaces/ITradePair.sol#127) (function)
ITradePair.setTotalAssetAmountLimit(uint256).totalAssetAmountLimit (src/interfaces/ITradePair.sol#191) shadows:
        - ITradePair.totalAssetAmountLimit() (src/interfaces/ITradePair.sol#129) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Pragma version^0.8.0 (src/interfaces/IController.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IFeeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILiquidityPoolAdapter.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAdapter.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradePair.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUserManager.sol#3) allows old versions
Pragma version0.8.17 (src/shared/UnlimitedOwnable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/sys-controller/Controller.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable UnlimitedOwnable.constructor(IUnlimitedOwner)._unlimitedOwner (src/shared/UnlimitedOwnable.sol#20) is too similar to Controller.constructor(IUnlimitedOwner).unlimitedOwner_ (src/sys-controller/Controller.sol#43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

UnlimitedOwner.sol

```
Pragma version^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3) allows old versions
Pragma version0.8.17 (src/sys-controller/UnlimitedOwner.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

TradeManagerOrders.sol

```
TradeManager._updateContracts(UpdateData[]).i (src/trade-manager/TradeManager.sol#361) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).name (src/interfaces/ITradePair.sol#166) shadows:
        - ITradePair.name() (src/interfaces/ITradePair.sol#89) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).collateral (src/interfaces/ITradePair.sol#167) shadows:
        - ITradePair.collateral() (src/interfaces/ITradePair.sol#91) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).priceFeedAdapter (src/interfaces/ITradePair.sol#169) shadows:
        - ITradePair.priceFeedAdapter() (src/interfaces/ITradePair.sol#97) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).liquidityPoolAdapter (src/interfaces/ITradePair.sol#170) shadows:
        - ITradePair.liquidityPoolAdapter() (src/interfaces/ITradePair.sol#99) (function)
ITradePair.setLiquidatorReward(uint256).liquidatorReward (src/interfaces/ITradePair.sol#179) shadows:
        - ITradePair.liquidatorReward() (src/interfaces/ITradePair.sol#119) (function)
ITradePair.setMinLeverage(uint128).minLeverage (src/interfaces/ITradePair.sol#181) shadows:
        - ITradePair.minLeverage() (src/interfaces/ITradePair.sol#123) (function)
ITradePair.setMaxLeverage(uint128).maxLeverage (src/interfaces/ITradePair.sol#183) shadows:
        - ITradePair.maxLeverage() (src/interfaces/ITradePair.sol#121) (function)
ITradePair.setMinMargin(uint256).minMargin (src/interfaces/ITradePair.sol#185) shadows:
        - ITradePair.minMargin() (src/interfaces/ITradePair.sol#125) (function)
ITradePair.setVolumeLimit(uint256).volumeLimit (src/interfaces/ITradePair.sol#187) shadows:
        - ITradePair.volumeLimit() (src/interfaces/ITradePair.sol#127) (function)
ITradePair.setTotalAssetAmountLimit(uint256).totalAssetAmountLimit (src/interfaces/ITradePair.sol#191) shadows:
        - ITradePair.totalAssetAmountLimit() (src/interfaces/ITradePair.sol#129) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

TradeManager.onlyActiveTradePair(address) (src/trade-manager/TradeManager.sol#377-380) has external calls inside a loop: controller.checkTradePairActive(tradePair_) (src/trade-manager/TradeManager.sol#378)
TradeManager._tryLiquidatePosition(address,uint256,address) (src/trade-manager/TradeManager.sol#167-178) has external calls inside a loop: ITradePair(tradePair_).liquidatePosition(maker_,positionId_) (src/trade-manager/TradeManager.sol#172-177)
TradeManager.positionIsLiquidatable(address,uint256) (src/trade-manager/TradeManager.sol#256-258) has external calls inside a loop: ITradePair(tradePair_).positionIsLiquidatable(positionId_) (src/trade-manager/TradeManager.sol#257)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in TradeManager._addMarginToPosition(AddMarginToPositionParams,address) (src/trade-manager/TradeManager.sol#105-114):
        External calls:
        - ITradePair(params_.tradePair).collateral().safeTransferFrom(maker_,address(params_.tradePair),params_.addedMargin) (src/trade-manager/TradeManager.sol#107-109)
        - ITradePair(params_.tradePair).addMarginToPosition(maker_,params_.positionId,params_.addedMargin) (src/trade-manager/TradeManager.sol#111)
        Event emitted after the call(s):
        - MarginAddedToPosition(params_.tradePair,params_.positionId,params_.addedMargin) (src/trade-manager/TradeManager.sol#113)
Reentrancy in TradeManager._closePosition(ClosePositionParams,address) (src/trade-manager/TradeManager.sol#74-77):
        External calls:
        - ITradePair(params_.tradePair).closePosition(maker_,params_.positionId) (src/trade-manager/TradeManager.sol#75)
        Event emitted after the call(s):
        - PositionClosed(params_.tradePair,params_.positionId) (src/trade-manager/TradeManager.sol#76)
Reentrancy in TradeManager._extendPosition(ExtendPositionParams,address) (src/trade-manager/TradeManager.sol#121-132):
        External calls:
        - ITradePair(params_.tradePair).collateral().safeTransferFrom(maker_,address(params_.tradePair),params_.addedMargin) (src/trade-manager/TradeManager.sol#123-125)
        - ITradePair(params_.tradePair).extendPosition(maker_,params_.positionId,params_.addedMargin,params_.addedLeverage) (src/trade-manager/TradeManager.sol#127-129)
        Event emitted after the call(s):
        - PositionExtended(params_.tradePair,params_.positionId,params_.addedMargin,params_.addedLeverage) (src/trade-manager/TradeManager.sol#131)
Reentrancy in TradeManager._extendPositionToLeverage(ExtendPositionToLeverageParams,address) (src/trade-manager/TradeManager.sol#139-143):
        External calls:
        - ITradePair(params_.tradePair).extendPositionToLeverage(maker_,params_.positionId,params_.targetLeverage) (src/trade-manager/TradeManager.sol#140)
        Event emitted after the call(s):
        - PositionExtendedToLeverage(params_.tradePair,params_.positionId,params_.targetLeverage) (src/trade-manager/TradeManager.sol#142)
Reentrancy in TradeManager._openPosition(OpenPositionParams,address) (src/trade-manager/TradeManager.sol#54-66):
        External calls:
        - ITradePair(params_.tradePair).collateral().safeTransferFrom(maker_,address(params_.tradePair),params_.margin) (src/trade-manager/TradeManager.sol#55)
        - userManager.setUserReferrer(maker_,params_.referrer) (src/trade-manager/TradeManager.sol#57)
        - id = ITradePair(params_.tradePair).openPosition(maker_,params_.margin,params_.leverage,params_.isShort,params_.whitelabelAddress) (src/trade-manager/TradeManager.sol#59-61)
        Event emitted after the call(s):
        - PositionOpened(params_.tradePair,id) (src/trade-manager/TradeManager.sol#63)
Reentrancy in TradeManager._partiallyClosePosition(PartiallyClosePositionParams,address) (src/trade-manager/TradeManager.sol#84-87):
        External calls:
        - ITradePair(params_.tradePair).partiallyClosePosition(maker_,params_.positionId,params_.proportion) (src/trade-manager/TradeManager.sol#85)
        Event emitted after the call(s):
        - PositionPartiallyClosed(params_.tradePair,params_.positionId,params_.proportion) (src/trade-manager/TradeManager.sol#86)
Reentrancy in TradeManager._removeMarginFromPosition(RemoveMarginFromPositionParams,address) (src/trade-manager/TradeManager.sol#94-98):
        External calls:
        - ITradePair(params_.tradePair).removeMarginFromPosition(maker_,params_.positionId,params_.removedMargin) (src/trade-manager/TradeManager.sol#95)
        Event emitted after the call(s):
        - MarginRemovedFromPosition(params_.tradePair,params_.positionId,params_.removedMargin) (src/trade-manager/TradeManager.sol#97)
Reentrancy in TradeManagerOrders._transferOrderReward(address,address,address) (src/trade-manager/TradeManagerOrders.sol#414-422):
        External calls:
        - collateral.safeTransferFrom(from_,to_,orderReward) (src/trade-manager/TradeManagerOrders.sol#418)
        Event emitted after the call(s):
        - OrderRewardTransfered(address(collateral),from_,to_,orderReward) (src/trade-manager/TradeManagerOrders.sol#421)
Reentrancy in TradeManager._tryLiquidatePosition(address,uint256,address) (src/trade-manager/TradeManager.sol#167-178):
        External calls:
        - ITradePair(tradePair_).liquidatePosition(maker_,positionId_) (src/trade-manager/TradeManager.sol#172-177)
        Event emitted after the call(s):
        - PositionLiquidated(tradePair_,positionId_) (src/trade-manager/TradeManager.sol#173)
Reentrancy in TradeManager.batchLiquidatePositions(address[],uint256[][],bool,UpdateData[]) (src/trade-manager/TradeManager.sol#190-205):
        External calls:
        - _updateContracts(updateData_) (src/trade-manager/TradeManager.sol#197)
                - IUpdatable(updateData_[i].updatableContract).update(updateData_[i].data) (src/trade-manager/TradeManager.sol#367)
        - didLiquidate[i] = _batchLiquidatePositionsOfTradePair(tradePairs[i],positionIds[i],allowRevert,msg.sender) (src/trade-manager/TradeManager.sol#202-203)
                - ITradePair(tradePair_).liquidatePosition(maker_,positionId_) (src/trade-manager/TradeManager.sol#172-177)
        Event emitted after the call(s):
        - PositionLiquidated(tradePair_,positionId_) (src/trade-manager/TradeManager.sol#173)
                - didLiquidate[i] = _batchLiquidatePositionsOfTradePair(tradePairs[i],positionIds[i],allowRevert,msg.sender) (src/trade-manager/TradeManager.sol#202-203)
```

```
Reentrancy in TradeManager.liquidatePosition(address,uint256,UpdateData[]) (src/trade-manager/TradeManager.sol#153-160):
        External calls:
        - _updateContracts(updateData_) (src/trade-manager/TradeManager.sol#157)
            - IUpdatable(updateData_[i].updatableContract).update(updateData_[i].data) (src/trade-manager/TradeManager.sol#367)
        - ITradePair(tradePair_).liquidatePosition(msg.sender,positionId_) (src/trade-manager/TradeManager.sol#158)
        Event emitted after the call(s):
        - PositionLiquidated(tradePair_,positionId_) (src/trade-manager/TradeManager.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

TradeManager._verifyConstraints(address,Constraints,UsePrice) (src/trade-manager/TradeManager.sol#333-355) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(constraints_.deadline > block.timestamp,TradeManager::_verifyConstraints: Deadline passed) (src/trade-manager/TradeManager.sol#338)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Pragma version^0.8.0 (src/interfaces/IController.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IFeeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILLiquidityPoolAdapter.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAdapter.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradeManagerOrders.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradePair.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradeSignature.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUpdatable.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUserManager.sol#3) allows old versions
Pragma version0.8.17 (src/trade-manager/TradeManager.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/trade-manager/TradeManagerOrders.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/trade-manager/TradeSignature.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable TradeManager.liquidatePosition(address,uint256,UpdateData[]).positionId_ (src/trade-manager/TradeManager.sol#153) is too similar to ITradeManager.canLiquidatePositions(address[],uint256[][]).positionIds (src/interfaces/ITradeMan
ager.sol#268)
Variable TradeManager._tryLiquidatePosition(address,uint256,address).positionId_ (src/trade-manager/TradeManager.sol#167) is too similar to TradeManager.batchLiquidatePositions(address[],uint256[][],bool,UpdateData[]).positionIds (src/tr
ade-manager/TradeManager.sol#192)
Variable TradeManager.liquidatePosition(address,uint256,UpdateData[]).positionId_ (src/trade-manager/TradeManager.sol#153) is too similar to TradeManager._batchLiquidatePositionsOfTradePair(address,uint256[],bool,address).positionIds (sr
c/trade-manager/TradeManager.sol#216)
Variable TradeManager._tryLiquidatePosition(address,uint256,address).positionId_ (src/trade-manager/TradeManager.sol#167) is too similar to ITradeManager.canLiquidatePositions(address[],uint256[][]).positionIds (src/interfaces/ITradeMana
ger.sol#268)
Variable TradeManager._tryLiquidatePosition(address,uint256,address).positionId_ (src/trade-manager/TradeManager.sol#167) is too similar to TradeManager._batchLiquidatePositionsOfTradePair(address,uint256[],bool,address).positionIds (src
/trade-manager/TradeManager.sol#216)
Variable TradeManager.liquidatePosition(address,uint256,UpdateData[]).positionId_ (src/trade-manager/TradeManager.sol#153) is too similar to TradeManager.batchLiquidatePositions(address[],uint256[][],bool,UpdateData[]).positionIds (src/t
rade-manager/TradeManager.sol#192)
Variable TradeManager.liquidatePosition(address,uint256,UpdateData[]).positionId_ (src/trade-manager/TradeManager.sol#153) is too similar to ITradeManager.batchLiquidatePositions(address[],uint256[][],bool,UpdateData[]).positionIds (src/
interfaces/ITradeManager.sol#257)
Variable TradeManager.positionIsLiquidatable(address,uint256).positionId_ (src/trade-manager/TradeManager.sol#256) is too similar to ITradeManager.batchLiquidatePositions(address[],uint256[][],bool,UpdateData[]).positionIds (src/interfac
es/ITradeManager.sol#257)
Variable TradeManager.detailsOfPosition(address,uint256).positionId_ (src/trade-manager/TradeManager.sol#243) is too similar to TradeManager.batchLiquidatePositions(address[],uint256[][],bool,UpdateData[]).positionIds (src/trade-manager/
TradeManager.sol#192)
Variable TradeManager.positionIsLiquidatable(address,uint256).positionId_ (src/trade-manager/TradeManager.sol#256) is too similar to TradeManager.batchLiquidatePositions(address[],uint256[][],bool,UpdateData[]).positionIds (src/trade-man
ager/TradeManager.sol#192)
Variable TradeManager.detailsOfPosition(address,uint256).positionId_ (src/trade-manager/TradeManager.sol#243) is too similar to ITradeManager.canLiquidatePositions(address[],uint256[][]).positionIds (src/interfaces/ITradeManager.sol#268)
Variable TradeManager.positionIsLiquidatable(address,uint256).positionId_ (src/trade-manager/TradeManager.sol#256) is too similar to TradeManager._batchLiquidatePositionsOfTradePair(address,uint256[],bool,address).positionIds (src/trade-
manager/TradeManager.sol#216)
Variable TradeManager._tryLiquidatePosition(address,uint256,address).positionId_ (src/trade-manager/TradeManager.sol#167) is too similar to ITradeManager.batchLiquidatePositions(address[],uint256[][],bool,UpdateData[]).positionIds (src/i
nterfaces/ITradeManager.sol#257)
Variable TradeManager.detailsOfPosition(address,uint256).positionId_ (src/trade-manager/TradeManager.sol#243) is too similar to ITradeManager.batchLiquidatePositions(address[],uint256[][],bool,UpdateData[]).positionIds (src/interfaces/IT
radeManager.sol#257)
Variable TradeManager.positionIsLiquidatable(address,uint256).positionId_ (src/trade-manager/TradeManager.sol#256) is too similar to ITradeManager.canLiquidatePositions(address[],uint256[][]).positionIds (src/interfaces/ITradeManager.sol
#268)
Variable TradeManager.detailsOfPosition(address,uint256).positionId_ (src/trade-manager/TradeManager.sol#243) is too similar to TradeManager._batchLiquidatePositionsOfTradePair(address,uint256[],bool,address).positionIds (src/trade-manag
er/TradeManager.sol#216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

liquidatePosition(address,uint256,UpdateData[]) should be declared external:
        - TradeManager.liquidatePosition(address,uint256,UpdateData[]) (src/trade-manager/TradeManager.sol#153-160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TradePair.sol

Contract not supported by Slither.

TradePairHelper.sol

```
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).name (src/interfaces/ITradePair.sol#166) shadows:
        - ITradePair.name() (src/interfaces/ITradePair.sol#89) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).collateral (src/interfaces/ITradePair.sol#167) shadows:
        - ITradePair.collateral() (src/interfaces/ITradePair.sol#91) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).priceFeedAdapter (src/interfaces/ITradePair.sol#169) shadows:
        - ITradePair.priceFeedAdapter() (src/interfaces/ITradePair.sol#97) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).liquidityPoolAdapter (src/interfaces/ITradePair.sol#170) shadows:
        - ITradePair.liquidityPoolAdapter() (src/interfaces/ITradePair.sol#99) (function)
ITradePair.setLiquidatorReward(uint256).liquidatorReward (src/interfaces/ITradePair.sol#179) shadows:
        - ITradePair.liquidatorReward() (src/interfaces/ITradePair.sol#119) (function)
ITradePair.setMinLeverage(uint128).minLeverage (src/interfaces/ITradePair.sol#181) shadows:
        - ITradePair.minLeverage() (src/interfaces/ITradePair.sol#123) (function)
ITradePair.setMaxLeverage(uint128).maxLeverage (src/interfaces/ITradePair.sol#183) shadows:
        - ITradePair.maxLeverage() (src/interfaces/ITradePair.sol#121) (function)
ITradePair.setMinMargin(uint256).minMargin (src/interfaces/ITradePair.sol#185) shadows:
        - ITradePair.minMargin() (src/interfaces/ITradePair.sol#125) (function)
ITradePair.setVolumeLimit(uint256).volumeLimit (src/interfaces/ITradePair.sol#187) shadows:
        - ITradePair.volumeLimit() (src/interfaces/ITradePair.sol#127) (function)
ITradePair.setTotalAssetAmountLimit(uint256).totalAssetAmountLimit (src/interfaces/ITradePair.sol#191) shadows:
        - ITradePair.totalAssetAmountLimit() (src/interfaces/ITradePair.sol#129) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

TradePairHelper.positionIdsOf(address,ITradePair[]) (src/trade-pair/TradePairHelper.sol#15-24) has external calls inside a loop: positionIds[i] = tradePairs_[i].positionIdsOf(maker_) (src/trade-pair/TradePairHelper.sol#22)
TradePairHelper.positionDetailsOf(address,ITradePair[]) (src/trade-pair/TradePairHelper.sol#32-45) has external calls inside a loop: positionIds = tradePairs_[i].positionIdsOf(maker_) (src/trade-pair/TradePairHelper.sol#39)
TradePairHelper.positionDetailsOf(address,ITradePair[]) (src/trade-pair/TradePairHelper.sol#32-45) has external calls inside a loop: positionDetails[i][j] = tradePairs_[i].detailsOfPosition(positionIds[j]) (src/trade-pair/TradePairHelper
.sol#42)
TradePairHelper.pricesOf(ITradePair[]) (src/trade-pair/TradePairHelper.sol#52-59) has external calls inside a loop: (minPrice,maxPrice) = tradePairs_[i].getCurrentPrices() (src/trade-pair/TradePairHelper.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Pragma version^0.8.0 (src/interfaces/IController.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IFeeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILLiquidityPoolAdapter.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAdapter.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradePair.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradePairHelper.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUserManager.sol#3) allows old versions
Pragma version0.8.17 (src/trade-pair/TradePairHelper.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

UserManager.sol

```
UserManager._getTimeIndexAndPosition(uint256) (src/user-manager/UserManager.sol#444-451) uses a weak PRNG: "position = dayaFromUnix % DAYS_IN_WORD (src/user-manager/UserManager.sol#449)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).name (src/interfaces/ITradePair.sol#166) shadows:
        - ITradePair.name() (src/interfaces/ITradePair.sol#89) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).collateral (src/interfaces/ITradePair.sol#167) shadows:
        - ITradePair.collateral() (src/interfaces/ITradePair.sol#91) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).priceFeedAdapter (src/interfaces/ITradePair.sol#169) shadows:
        - ITradePair.priceFeedAdapter() (src/interfaces/ITradePair.sol#97) (function)
ITradePair.initialize(string,IERC20Metadata,uint256,IPriceFeedAdapter,ILiquidityPoolAdapter).liquidityPoolAdapter (src/interfaces/ITradePair.sol#170) shadows:
        - ITradePair.liquidityPoolAdapter() (src/interfaces/ITradePair.sol#99) (function)
ITradePair.setLiquidatorReward(uint256).liquidatorReward (src/interfaces/ITradePair.sol#179) shadows:
        - ITradePair.liquidatorReward() (src/interfaces/ITradePair.sol#119) (function)
ITradePair.setMinLeverage(uint128).minLeverage (src/interfaces/ITradePair.sol#181) shadows:
        - ITradePair.minLeverage() (src/interfaces/ITradePair.sol#123) (function)
ITradePair.setMaxLeverage(uint128).maxLeverage (src/interfaces/ITradePair.sol#183) shadows:
        - ITradePair.maxLeverage() (src/interfaces/ITradePair.sol#121) (function)
ITradePair.setMinMargin(uint256).minMargin (src/interfaces/ITradePair.sol#185) shadows:
        - ITradePair.minMargin() (src/interfaces/ITradePair.sol#125) (function)
ITradePair.setVolumeLimit(uint256).volumeLimit (src/interfaces/ITradePair.sol#187) shadows:
        - ITradePair.volumeLimit() (src/interfaces/ITradePair.sol#127) (function)
ITradePair.setTotalAssetAmountLimit(uint256).totalAssetAmountLimit (src/interfaces/ITradePair.sol#191) shadows:
        - ITradePair.totalAssetAmountLimit() (src/interfaces/ITradePair.sol#129) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

UserManager.getUserManualTier(address) (src/user-manager/UserManager.sol#245-251) uses timestamp for comparisons
        Dangerous comparisons:
        - manualUserTiers[user_].validUntil >= block.timestamp (src/user-manager/UserManager.sol#246)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version^0.8.0 (src/interfaces/IController.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IFeeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ILiquidityPoolAdapter.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAdapter.sol#2) allows old versions
Pragma version^0.8.0 (src/interfaces/IPriceFeedAggregator.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradeManager.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/ITradePair.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUnlimitedOwner.sol#3) allows old versions
Pragma version^0.8.0 (src/interfaces/IUserManager.sol#3) allows old versions
Pragma version0.8.17 (src/shared/UnlimitedOwnable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (src/user-manager/UserManager.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable UserManager.getUserVolumeTier(address)._feeVolumes (src/user-manager/UserManager.sol#210) is too similar to UserManager.initialize(uint8[7],uint32[6]).feeVolumes_ (src/user-manager/UserManager.sol#131)
Variable UnlimitedOwnable.constructor(IUnlimitedOwner)._unlimitedOwner (src/shared/UnlimitedOwnable.sol#20) is too similar to UserManager.constructor(IUnlimitedOwner,IController,ITradeManager).unlimitedOwner_ (src/user-manager/UserManage
r.sol#121)
Variable UserManager.getUserVolumeTier(address)._feeVolumes (src/user-manager/UserManager.sol#210) is too similar to UserManager.setFeeVolumes(uint256[],uint32[]).feeVolumes_ (src/user-manager/UserManager.sol#372)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

initialize(uint8[7],uint32[6]) should be declared external:
        - UserManager.initialize(uint8[7],uint32[6]) (src/user-manager/UserManager.sol#131-141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- All the reentrancies flagged by Slither were checked individually and are false positives.
- No major issues found by Slither.

6.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with the detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

FeeManager.sol

Report for src/fee-manager/FeeManager.sol
https://dashboard.mythx.io/#/console/analyses/e474d385-6ad6-4a71-8488-538e9789ab65

Line	SWC Title	Severity	Short Description
228	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
228	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
228	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
229	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
229	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
230	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
230	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
251	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
251	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
251	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
252	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
252	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
253	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
253	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
271	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
271	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
322	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
322	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
327	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
337	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
337	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
342	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
342	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
345	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
346	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
354	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
357	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
357	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
359	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered

362	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
362	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
364	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered

LiquidityPool.sol

Report for src/liquidity-pools/LiquidityPool.sol
https://dashboard.mythx.io/#/console/analyses/e6e92830-81d7-4a90-a09c-219a49a5a742

Line	SWC Title	Severity	Short Description
68	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
90	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
153	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
168	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
169	(SWC-110) Assert Violation	Unknown	Out of bounds array access
196	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
205	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
214	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
287	(SWC-110) Assert Violation	Unknown	Out of bounds array access
292	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
293	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
302	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
305	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
363	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
363	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
363	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
387	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
388	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
392	(SWC-110) Assert Violation	Unknown	Out of bounds array access
396	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
397	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
406	(SWC-110) Assert Violation	Unknown	Out of bounds array access
409	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
409	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
409	(SWC-110) Assert Violation	Unknown	Out of bounds array access
423	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
435	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
450	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
466	(SWC-110) Assert Violation	Unknown	Out of bounds array access
470	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
471	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
473	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
501	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
505	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
505	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
505	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
509	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
520	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
520	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
520	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
521	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
521	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
521	(SWC-110) Assert Violation	Unknown	Out of bounds array access

522	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
523	(SWC-110) Assert Violation	Unknown	Out of bounds array access
523	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
526	(SWC-110) Assert Violation	Unknown	Out of bounds array access
526	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
526	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
526	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
555	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
556	(SWC-110) Assert Violation	Unknown	Out of bounds array access
558	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
558	(SWC-110) Assert Violation	Unknown	Out of bounds array access
559	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
559	(SWC-110) Assert Violation	Unknown	Out of bounds array access
592	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
592	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
594	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
594	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
610	(SWC-110) Assert Violation	Unknown	Out of bounds array access
614	(SWC-110) Assert Violation	Unknown	Out of bounds array access
617	(SWC-110) Assert Violation	Unknown	Out of bounds array access
618	(SWC-110) Assert Violation	Unknown	Out of bounds array access

LiquidityPoolAdapter.sol

Report for src/liquidity-pools/LiquidityPoolAdapter.sol
https://dashboard.mythx.io/#/console/analyses/445843dd-1b60-4ca4-848c-7c57a15bf648

Line	SWC Title	Severity	Short Description
32	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
84	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
86	(SWC-110) Assert Violation	Unknown	Out of bounds array access
91	(SWC-110) Assert Violation	Unknown	Out of bounds array access
94	(SWC-110) Assert Violation	Unknown	Out of bounds array access
130	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
131	(SWC-110) Assert Violation	Unknown	Out of bounds array access
132	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
132	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
132	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
132	(SWC-110) Assert Violation	Unknown	Out of bounds array access
143	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
143	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
160	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
161	(SWC-110) Assert Violation	Unknown	Out of bounds array access
162	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
162	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
162	(SWC-110) Assert Violation	Unknown	Out of bounds array access
163	(SWC-110) Assert Violation	Unknown	Out of bounds array access
163	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
167	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
167	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
175	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
176	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
176	(SWC-110) Assert Violation	Unknown	Out of bounds array access
176	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered

177	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
179	(SWC-110) Assert Violation	Unknown	Out of bounds array access
222	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
223	(SWC-110) Assert Violation	Unknown	Out of bounds array access
224	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
224	(SWC-110) Assert Violation	Unknown	Out of bounds array access
224	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
225	(SWC-110) Assert Violation	Unknown	Out of bounds array access
225	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
230	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
231	(SWC-110) Assert Violation	Unknown	Out of bounds array access
232	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
238	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
238	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
238	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
239	(SWC-110) Assert Violation	Unknown	Out of bounds array access
239	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
239	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
240	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
243	(SWC-110) Assert Violation	Unknown	Out of bounds array access
247	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
248	(SWC-110) Assert Violation	Unknown	Out of bounds array access
248	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
248	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered

ChainlinkUsdPriceFeed.sol

No issues found by MythX.

ChainlinkUsdPriceFeedPrevious.sol

No issues found by MythX.

PriceFeedAdapter.sol

Report for price-feed/PriceFeedAdapter.sol
https://dashboard.mythx.io/#/console/analyses/36a20248-69f1-4ec9-84f0-5e589b16d855

Line	SWC Title	Severity	Short Description
15	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
17	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

PriceFeedAggregator.sol

Report for price-feed/PriceFeedAggregator.sol
https://dashboard.mythx.io/#/console/analyses/65b16c02-3deb-4963-900b-481ac201da26

Line	SWC Title	Severity	Short Description
21	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
62	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
62	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
62	(SWC-110) Assert Violation	Unknown	Out of bounds array access
98	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
99	(SWC-110) Assert Violation	Unknown	Out of bounds array access

UnlimitedPriceFeed.sol

Report for src/price-feed/UnlimitedPriceFeed.sol
https://dashboard.mythx.io/#/console/analyses/0ee57ca4-0a82-4084-8dec-a5b74ecc8def

Line	SWC Title	Severity	Short Description
41	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
44	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
44	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
120	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
120	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
123	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
124	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

UnlimitedPriceFeedAdapter.sol

Report for src/price-feed/UnlimitedPriceFeedAdapter.sol
https://dashboard.mythx.io/#/console/analyses/e3e23087-c5ed-49e2-b9a6-aa78d17b117c

Line	SWC Title	Severity	Short Description
73	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
74	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
96	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
96	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
116	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
116	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
144	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
144	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
170	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
170	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
212	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
212	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
215	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
215	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
222	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
222	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered

Controller.sol

No issues found by MythX.

UnlimitedOwner.sol

No issues found by MythX.

TradeManagerOrders.sol

Report for src/trade-manager/TradeManager.sol
https://dashboard.mythx.io/#/console/analyses/aa40ddb3-321e-4c8e-8c6d-d926fd4280a6

Line	SWC Title	Severity	Short Description
201	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
202	(SWC-110) Assert Violation	Unknown	Out of bounds array access
203	(SWC-110) Assert Violation	Unknown	Out of bounds array access
222	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
223	(SWC-110) Assert Violation	Unknown	Out of bounds array access
224	(SWC-110) Assert Violation	Unknown	Out of bounds array access
227	(SWC-110) Assert Violation	Unknown	Out of bounds array access
279	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered

281	(SWC-110) Assert Violation	Unknown	Out of bounds array access
297	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
298	(SWC-110) Assert Violation	Unknown	Out of bounds array access
361	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
363	(SWC-110) Assert Violation	Unknown	Out of bounds array access
367	(SWC-110) Assert Violation	Unknown	Out of bounds array access

TradePair.sol

Report for src/trade-pair/TradePair.sol
https://dashboard.mythx.io/#/console/analyses/26d4bef1-b445-42ef-b3e8-e536c86e11b7

Line	SWC Title	Severity	Short Description
33	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
33	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
34	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
36	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
102	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
109	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
202	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
206	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
206	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
232	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
271	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
279	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
280	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
331	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
332	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
333	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
334	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
338	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
339	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
394	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
394	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
453	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
454	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
456	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
571	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
577	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
581	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
593	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
602	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
611	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
645	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
647	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
647	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
670	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
671	(SWC-110) Assert Violation	Unknown	Out of bounds array access
672	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
672	(SWC-110) Assert Violation	Unknown	Out of bounds array access
672	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
700	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
739	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
752	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
774	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
798	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
811	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
1117	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
1128	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
1131	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

TradePairHelper.sol

Report for src/trade-pair/TradePairHelper.sol
https://dashboard.mythx.io/#/console/analyses/91ec3009-61e8-4d20-a435-47b3fc3d9f15

Line	SWC Title	Severity	Short Description
8	(SWC-123) Requirement Violation	Low	Requirement violation.
55	(SWC-123) Requirement Violation	Low	Requirement violation.
55	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.

UserManager.sol

Report for src/user-manager/UserManager.sol
https://dashboard.mythx.io/#/console/analyses/f3fb6394-a0ab-4b6a-94d6-37b3272c4c7d

Line	SWC Title	Severity	Short Description
135	(SWC-110) Assert Violation	Unknown	Out of bounds array access
137	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
138	(SWC-110) Assert Violation	Unknown	Out of bounds array access
138	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
139	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
139	(SWC-110) Assert Violation	Unknown	Out of bounds array access
260	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
265	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
355	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
356	(SWC-110) Assert Violation	Unknown	Out of bounds array access
375	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
376	(SWC-110) Assert Violation	Unknown	Out of bounds array access
389	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
391	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
393	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
395	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
397	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
399	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
436	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
436	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
446	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
448	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
449	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered

- Integer Overflows and Underflows flagged by MythX are false positives, as the contracts are using Solidity 0.8.17 version. After the Solidity version 0.8.0 Arithmetic operations revert to underflow and overflow by default.
- Assert violations are false positives.
- No major issues were found by MythX.

PULL REQUEST REVIEW OVERVIEW

In the PR#121 review carried out between April 5th and April 10th, the following commits were examined by Halborn:

- upgrade foundry
- remove deprecated AggregatorV2Interface
- update snapshot
- change i++ to ++i to save gas
- remove unneeded initialization of integer variables to 0
- add test for minMargin to removeMargin
- add test for minLeverage at addMargin
- add check to prevent TradePair depletion
- forge update
- feature: enable borrow fee rate of zero
- add total volume limit
- HAL17 optimize detailsOfPosition
- HAL09: Implement double-step ownership transfer
- HAL04: add IEP712 to PFA Updater
- WIP make FeeManager uninitializable adapt tests to deployment script
- HAL10: Add disableInitializers to all initializable contracts
- HAL11: Add storage gaps to upgradeable contracts
- remove test deployment artifacts from repo
- HAL02: Add validation checks to openPosition
- HAL06: require asset decimals of <= 18 at LiquidityPool
- HAL12: remove positionIdsOf from TradePair
- CR: base closeFee on volume
- CR: Add event CollectedEarlyWithdrawalFee
- change deployment script to use totalVolumeLimit
- feat: make price feed adapter upgradeable
- adapt tests to use Proxy
- adapt deployment script to deploy proxy for PFA
- FIX: Bug at TradeManagerOrders
- forge install: solarray
- add detailsOfPositions to TradePairHelper
- second attempt to add arbitrary price decimals
- make test pass and set network correctly
- make all tests pass
- add Stresstest

- add first attempt with PositionMaths calling TradePair
- refactor: remove asset_decimals from smart contracts
- remove assetDecimals from PositionMaths
- refactor: UnlimitedPriceFeedAdapter
- add StressTest and make tests pass
- add canLiquidatePositionsAtPrices
- use arbitrum block instead of block.number
- feature: improve Upgradeability of UnlimitedPriceFeedAdapter
- HAL: remove disableInitializer from all contracts except UnlimitedOwner
- HAL03: add security checks for chainlink price feeds
- change calculation of liquidation price
- rename decimals
- add ArbSys to Position
- correct typo
- add prices array length check to canLiquidatePositionsAtPrice
- add comment to explain totalVolumeLimit to TradeManager
- remove +1 from maxLeverage check at TradePair
- forge fmt
- separate current from collcted fee amount
- add realized fee amounts to event RealizedPnL
- add funding and borrow fee amount to positionDetails
- adapt tokenMinter and mockToken
- distuingish deployment scripts for deployment tests
- add script DeployArbitrumGoerli
- add correct pricefeeds
- add configure function
- add configuration variables
- change MAX_DEVIATION
- change deployment script
- correct typo
- remove solarray as forge library
- add require cannot close a liquidatable position
- update realizedFundingFeeAmount

As a result of the audit on these commits, the HAL-11 and HAL-19 findings were identified and added to the report.

MANUAL TESTING

8.1 MANUAL TEST METHODOLOGY

The scope of the manual tests performed covered the position structure and fee values. They were tested to identity the conditions under which positions can be liquidated. The behavior of the EIP1271 integration was tested to verify the impact of a malicious Signer Contract interacting with the protocol.

test > mocks > MockFakeSigner.sol > ...

report | graph (this) | graph | inheritance | parse | flatten | funcSigs | uml | draw.io

```
1  // SPDX-License-Identifier: Unlicensed
2  pragma solidity ^0.8.17;
3
4  import "@openzeppelin/contracts/interfaces/IERC1271.sol";
5  import "@openzeppelin/contracts/interfaces/IERC20.sol";
6
   UnitTest stub | dependencies | uml | draw.io
7  contract FakeSigner {
8      address internal _owner;
9      bytes4 internal constant EIP1271_MAGIC_VALUE = IERC1271.isValidSignature.selector;
10
       ftrace
11     constructor() {
12         _owner == msg.sender;
13     }
14
       ftrace | funcSig
15     function isValidSignature(bytes32, bytes memory) external pure returns (bytes4) {
16         return EIP1271_MAGIC_VALUE;
17     }
18
       ftrace | funcSig
19     function enable(address _token⬆, address _origin⬆) external onlyOwner {
⚠ 20       IERC20(_token⬆).approve(_origin⬆, IERC20(_token⬆).balanceOf(address(this)));
21     }
22
23     modifier onlyOwner() {
24         require(_owner == msg.sender);
25         _;
26     }
27 }
28
```

test > TestClosingFees.t.sol > ...

report | graph (this) | graph | inheritance | parse | flatten | funcSigs | uml | draw.io

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity 0.8.17;

/**
 *  forge test -vvvv --match-contract FullTestClosingFees --fork-url https://rpc.ankr.com/eth
 */

import "forge-std/Test.sol";
import "src/price-feed/ChainlinkUsdPriceFeed.sol";
import "test/mocks/MockPriceFeedAdapter.sol";
import "src/interfaces/IPriceFeedAdapter.sol";
import "src/interfaces/IController.sol";
import "src/trade-pair/TradePair.sol";
import "src/liquidity-pools/LiquidityPool.sol";
import "src/fee-manager/FeeManager.sol";
import "src/trade-manager/TradeManagerOrders.sol";
import "src/liquidity-pools/LiquidityPoolAdapter.sol";
import "src/sys-controller/Controller.sol";
import "src/sys-controller/UnlimitedOwner.sol";
import "src/user-manager/UserManager.sol";
import "@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol";
import "test/mocks/MockFakeSigner.sol";

UnitTest stub | dependencies | uml | draw.io
contract FullTestClosingFees is Test {
    using PositionMaths for Position;

    // Contracts
    Controller public contract_Controller;
    UnlimitedOwner public contract_UnlimitedOwner;
    UserManager public contract_userManager;
    TradeManagerOrders public contract_tradeManager;
    FeeManager public contract_feeManager;
    IERC20Metadata public contract_Collateral;
    LiquidityPool public contract_liquidityPool0;
    LiquidityPool public contract_liquidityPool1;
    LiquidityPoolAdapter public contract_liquidityPoolAdapter;
    MockPriceFeedAdapter public contract_priceFeedAdapter;
    TradePair public contract_tradePair0;
    FakeSigner fakeSigner;
    UpdateData[] updateData;
```

```solidity
    // Constants
    address internal constant WHITELABEL_ADDRESS = address(9);
    address internal constant STAKERS_ADDRESS = address(10);
    address internal constant DEV_ADDRESS = address(11);
    address internal constant INSURANCE_ADDRESS = address(12);
    address constant BACKEND = 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720;
    address constant UNLIMITED_OWNER = address(9999);

    // Users/ERCs
    IERC20Metadata contract_DAI = IERC20Metadata(0x6B175474E89094C44Da98b954EedeAC495271d0F);
    IERC20Metadata contract_USDC = IERC20Metadata(0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48);
    address owner = 0x0000000000000000000000000000000000000100;
    address user1 = 0x0000000000000000000000000000000000000101;
    address user2 = 0x0000000000000000000000000000000000000102;
    address user3 = 0x0000000000000000000000000000000000000103;
    address user4 = 0x0000000000000000000000000000000000000104;
```

```
71  function setUp() public {
72      console.log("\n\nsetUp()");
73      console.log(
74          "_____________________________________________________________________________________________"
75      );
76      vm.startPrank(owner);
77      // Deploying UnlimitedOwner
78      contract_UnlimitedOwner = new UnlimitedOwner();
79      contract_UnlimitedOwner.initialize();
80      // Deploying Controller contract -> constructor(IUnlimitedOwner unlimitedOwner_) UnlimitedOwnable(unlimitedOwner_) {}
81      contract_Controller = new Controller(IUnlimitedOwner(contract_UnlimitedOwner));
82      contract_Controller.addOrderExecutor(BACKEND);
83      // Precompute TradeManager address
84      address tradeManager_ = computeCreateAddress(address(owner), vm.getNonce(address(owner)) + 1);
85      // Deploying UserManager contract -> constructor(IUnlimitedOwner unlimitedOwner_, IController controller_, ITradeManager tradeManager_)
86      contract_userManager =
87          new UserManager(contract_UnlimitedOwner, contract_Controller, ITradeManager(tradeManager_));
88      // To simplify tests, we use the same fee sizes for the first two tiers
89      uint8[7] memory feeSizes = [10, 10, 8, 7, 6, 5, 4];
90      uint32[6] memory volumes = [1_000_000, 10_000_000, 100_000_000, 250_000_000, 500_000_000, 1_000_000_000];
91      contract_userManager.initialize(feeSizes, volumes);
92      // Deploying TradeManager -> constructor(IController controller_, IUserManager userManager_)
93      contract_tradeManager = new TradeManagerOrders(contract_Controller, contract_userManager);
94      // Deploying FeeManager -> constructor(IUnlimitedOwner unlimitedOwner_, IController controller_, IUserManager userManager_)
95      contract_feeManager = new FeeManager(contract_UnlimitedOwner, contract_Controller, contract_userManager);
96      contract_feeManager.initialize(
97          10_00, // 10% max payout
98          STAKERS_ADDRESS,
99          DEV_ADDRESS,
100         INSURANCE_ADDRESS
101     );
102
103     // Collateral
104     contract_Collateral = contract_DAI;
105
106     // Deploying LiquidityPool contract -> constructor(IUnlimitedOwner unlimitedOwner_, IERC20Metadata collateral_, IController controller_)
107     contract_liquidityPool0 =
108     new LiquidityPool(IUnlimitedOwner(contract_UnlimitedOwner), contract_Collateral, IController(contract_Controller));
```

```
119     contract_liquidityPool0.initialize("DAIPool", "DAIP", 86400, 5_00, 0, 0);
120     contract_Controller.addLiquidityPool(address(contract_liquidityPool0));
121     contract_liquidityPool0.addPool(86400, 100_00);
122
123     contract_liquidityPool1 =
124     new LiquidityPool(IUnlimitedOwner(contract_UnlimitedOwner), contract_Collateral, IController(contract_Controller));
125     contract_liquidityPool1.initialize("DAIPool", "DAIP", 86400, 5_00, 0, 0);
126     contract_Controller.addLiquidityPool(address(contract_liquidityPool1));
127     contract_liquidityPool1.addPool(86400, 100_00);
128
129     // LiquidityPoolConfig
130     LiquidityPoolConfig[] memory liquidityConfig = new LiquidityPoolConfig[](2);
131     liquidityConfig[0] = LiquidityPoolConfig(address(contract_liquidityPool0), uint96(FULL_PERCENT));
132     liquidityConfig[1] = LiquidityPoolConfig(address(contract_liquidityPool1), uint96(FULL_PERCENT));
133     // Deploying LiquidityPoolAdapter
134     contract_liquidityPoolAdapter =
135     new LiquidityPoolAdapter(contract_UnlimitedOwner, contract_Controller, address(contract_feeManager), contract_Collateral);
136     contract_liquidityPoolAdapter.initialize(FULL_PERCENT, liquidityConfig);
137     contract_Controller.addLiquidityPoolAdapter(address(contract_liquidityPoolAdapter));
138
139     // Deploying PriceFeedAdapter
140     uint256 ASSET_DECIMALS = 18;
141     contract_priceFeedAdapter =
142         new MockPriceFeedAdapter("Mock Price Feed Adapter", ASSET_DECIMALS, contract_Collateral.decimals());
143     // int256 price = int256(2_000 * (10 ** contract_Collateral.decimals()));
144     // MIN PRICE = _999900000000000000
145     // MAX PRICE = 1_010000000000000000
146     contract_priceFeedAdapter.setMarkPrices(999900000000000000, 1_010000000000000000); // < <
147     contract_Controller.addPriceFeed(address(contract_priceFeedAdapter));
148
149     // Deploying TradePair
150     contract_tradePair0 = new TradePair(contract_UnlimitedOwner,
151         contract_tradeManager,
152         contract_userManager,
153         contract_feeManager
154     );
155     contract_tradePair0.initialize(
156         "X Coin Trade Pair",
157         contract_Collateral,
158         ASSET_DECIMALS,
159         contract_priceFeedAdapter,
160         contract_liquidityPoolAdapter
161     );
162     contract_Controller.addTradePair(address(contract_tradePair0));
```

```
164         // Configure
165         contract_tradePair0.setLiquidatorReward(5 * 10 ** 18);
166         contract_tradePair0.setMinLeverage(1_100_000); // 11 * 1_000_000 / 10 = 1_100_000
167         contract_tradePair0.setMaxLeverage(100_000_000); // 100 * 1_000_000;
168         contract_tradePair0.setMinMargin(100 * 10 ** 18); // 100 * 10 ** 18;
169         contract_tradePair0.setVolumeLimit(10_00_000_000 * 10 ** 18); // 10_00_000_000 * 10 ** 18;
170         contract_tradePair0.setBorrowFeeRate(100_000_000_000);
171         contract_tradePair0.setMaxFundingFeeRate(300_000_000_000); // 0.3%
172         contract_tradePair0.setMaxExcessRatio(2 * 1e14); // 2 * 1e14
173         contract_tradePair0.setFeeBufferFactor(250_000);
174         contract_tradePair0.setTotalAssetAmountLimit(1_000_000_000 * 10 ** 18); // 1_000_000_000 * 10 ** 18;
175
176         contract_Controller.setOrderRewardOfCollateral(address(contract_Collateral), 3 * 10 ** 18);
177
178         // Dealing some DAI
179         deal(address(contract_Collateral), user1, 200e18);
180         deal(address(contract_Collateral), user2, 200e18);
181         deal(address(contract_Collateral), user3, 200e18);
182         deal(address(contract_Collateral), user4, 200e18);
183         deal(address(contract_Collateral), ALICE, 210e18);
184         deal(address(contract_Collateral), BOB, 1100e18);
185
186         console.log("HALBORN::", contract_Collateral.balanceOf(address(fakeSigner)));
187
188         fakeSigner = new FakeSigner();
189         deal(address(contract_Collateral), address(fakeSigner), 10e18);
190
191         fakeSigner.enable(address(contract_Collateral), address(contract_tradeManager));
192
193         vm.stopPrank();
```

```
195     vm.startPrank(BOB);
196     // OPENING A POSITION AS BOB
197     OpenPositionParams memory _openPosParams =
198         OpenPositionParams(address(contract_tradePair0), 1000e18, 2_000_000, false, address(0), address(0));
199     Constraints memory _constraints =
200         Constraints(block.timestamp + 1 hours, 99900000000000000, 1_001000000000000000);
201     OpenPositionOrder memory _openPosOrder = OpenPositionOrder(_openPosParams, _constraints, 1);
202     bytes32 orderHash = contract_tradeManager.hash(_openPosOrder);
203     bytes memory orderSignature = _sign(BOB_PK, orderHash);
204     bytes32 positionSignatureHash = keccak256(orderSignature);
205     saveSignatureHash = positionSignatureHash;
206     console.log("orderHash");
207     console.logBytes32(orderHash);
208     console.log("orderSignature");
209     console.logBytes(orderSignature);
210     console.log("positionSignatureHash");
211     console.logBytes32(positionSignatureHash);
212     //UpdateData[] memory updateData;
213
214     // Approve
215     contract_Collateral.approve(address(contract_tradeManager), 1100e18);
216     vm.stopPrank();
217
218     vm.startPrank(BACKEND);
219
220     // OpenPosition
221     console.log("contract_DAI.balanceOf(BOB) -> %s", contract_DAI.balanceOf(BOB));
222     console.log(
223         "contract_DAI.balanceOf(contract_liquidityPool0) -> %s",
224         contract_DAI.balanceOf(address(contract_liquidityPool0))
225     );
226     console.log(
227         "contract_DAI.balanceOf(contract_liquidityPool1) -> %s",
228         contract_DAI.balanceOf(address(contract_liquidityPool1))
229     );
230     contract_tradeManager.openPositionViaSignature(_openPosOrder, updateData, BOB, orderSignature);
231     console.log("contract_DAI.balanceOf(BOB) -> %s", contract_DAI.balanceOf(BOB));
232     console.log(
233         "contract_DAI.balanceOf(contract_liquidityPool0) -> %s",
234         contract_DAI.balanceOf(address(contract_liquidityPool0))
235     );
236     console.log(
237         "contract_DAI.balanceOf(contract_liquidityPool1) -> %s",
238         contract_DAI.balanceOf(address(contract_liquidityPool1))
239     );
```

```
240         vm.stopPrank();
241
242         console.log("contract_Controller.address -> %s", address(contract_Controller));
243         console.log("contract_UnlimitedOwner.address -> %s", address(contract_UnlimitedOwner));
244         console.log("contract_userManager.address -> %s", address(contract_userManager));
245         console.log("contract_tradeManager.address -> %s", address(contract_tradeManager));
246         console.log("contract_feeManager.address -> %s", address(contract_feeManager));
247         console.log("contract_Collateral.address -> %s", address(contract_Collateral));
248         console.log("contract_liquidityPool0.address -> %s", address(contract_liquidityPool0));
249         console.log("contract_liquidityPool1.address -> %s", address(contract_liquidityPool1));
250         console.log("contract_liquidityPoolAdapter.address -> %s", address(contract_liquidityPoolAdapter));
251         console.log("contract_priceFeedAdapter.address -> %s", address(contract_priceFeedAdapter));
252         console.log("contract_tradePair0.address -> %s", address(contract_tradePair0));
253
254         _init_LiquidityPools();
255
256         // Deleting "fuzzingOutput.txt" file
257         string[] memory inputs = new string[](2);
258         inputs[0] = "rm";
259         inputs[1] = "output.txt";
260         //vm.ffi(inputs);
261     }
262
    ftrace | funcSig
263     function test_env_SetUpState() public view {
264         console.log("\n\ntest_env_SetUpState()");
265         console.log(
266             "_____________________________________________________________________________________________"
267         );
268         // Initialization
269         console.log("contract_tradePair0.maxLeverage() -> %s", contract_tradePair0.maxLeverage());
270         // DAI balances
271         console.log("contract_DAI.balanceOf(user1) -> %s", contract_DAI.balanceOf(user1));
272         console.log("contract_DAI.balanceOf(user2) -> %s", contract_DAI.balanceOf(user2));
273         console.log("contract_DAI.balanceOf(user3) -> %s", contract_DAI.balanceOf(user3));
274         console.log("contract_DAI.balanceOf(user4) -> %s", contract_DAI.balanceOf(user4));
275         console.log("contract_DAI.balanceOf(ALICE) -> %s", contract_DAI.balanceOf(ALICE));
276     }
```

```
281     function testFail_RemoveMargin() public {
282         uint256[] memory feeIndexes = new uint256[](7);
283         uint8[] memory feeSizes_ = new uint8[](7);
284
285         feeIndexes[0] = 0;
286         feeIndexes[1] = 1;
287         feeIndexes[2] = 2;
288         feeIndexes[3] = 3;
289         feeIndexes[4] = 4;
290         feeIndexes[5] = 5;
291         feeIndexes[6] = 6;
292
293         feeSizes_[0] = 1_00;
294         feeSizes_[1] = 1_00;
295         feeSizes_[2] = 1_00;
296         feeSizes_[3] = 1_00;
297         feeSizes_[4] = 1_00;
298         feeSizes_[5] = 1_00;
299         feeSizes_[6] = 1_00;
300
301         vm.startPrank(BOB);
302
303         bytes32 positionSignatureHash = saveSignatureHash;
304
305         RemoveMarginFromPositionOrder memory removeMarginOrder = RemoveMarginFromPositionOrder(
306             RemoveMarginFromPositionParams(
307                 address(contract_tradePair0),
308                 0,
309                 115792089237316195423570985008687907853269984665640564039456584007913129639935
310             ),
311             Constraints(block.timestamp + 1 hours, 999000000000000000, 1_001000000000000000),
312             positionSignatureHash,
313             0
314         );
315
316         bytes32 orderHash = contract_tradeManager.hashRemoveMarginFromPositionOrder(removeMarginOrder);
317         bytes memory signature = _sign(BOB_PK, orderHash);
318         vm.stopPrank();
319
320         skip(50 minutes);
321         vm.roll(block.number + 23);
322
323         contract_userManager.setFeeSizes(feeIndexes, feeSizes_);
324
```

```
325         vm.prank(BACKEND);
326         contract_tradeManager.removeMarginFromPositionViaSignature(removeMarginOrder, updateData, BOB, signature);
327     }
328
```

```
329     function test_closeFee() public {
330         uint256[] memory feeIndexes = new uint256[](7);
331         uint8[] memory feeSizes_ = new uint8[](7);
332
333         feeIndexes[0] = 0;
334         feeIndexes[1] = 1;
335         feeIndexes[2] = 2;
336         feeIndexes[3] = 3;
337         feeIndexes[4] = 4;
338         feeIndexes[5] = 5;
339         feeIndexes[6] = 6;
340
341         feeSizes_[0] = 1_00;
342         feeSizes_[1] = 1_00;
343         feeSizes_[2] = 1_00;
344         feeSizes_[3] = 1_00;
345         feeSizes_[4] = 1_00;
346         feeSizes_[5] = 1_00;
347         feeSizes_[6] = 1_00;
348
349         vm.startPrank(BOB);
350
351         bytes32 positionSignatureHash = saveSignatureHash;
352
353         ClosePositionOrder memory closePositionOrder = ClosePositionOrder(
354             ClosePositionParams(address(contract_tradePair0), type(uint256).max),
355             Constraints(block.timestamp + 1 hours, 99900000000000000, 1_00100000000000000),
356             positionSignatureHash,
357             0
358         );
359
360         bytes32 orderHash = contract_tradeManager.hash(closePositionOrder);
361         bytes memory signature = _sign(BOB_PK, orderHash);
362         vm.stopPrank();
363
364         skip(50 minutes);
365         vm.roll(block.number + 23);
366
367         contract_userManager.setFeeSizes(feeIndexes, feeSizes_);
368
369         vm.prank(BACKEND);
370         contract_tradeManager.closePositionViaSignature(closePositionOrder, updateData, BOB, signature);
```

```
372             vm.prank(BACKEND);
373             bytes memory fakeSignature;
374             contract_tradeManager.closePositionViaSignature(
375                 closePositionOrder, updateData, address(fakeSigner), fakeSignature
376             );
377         }
378
```

```
379     function _sign(uint256 signerPk_⬆, bytes32 dataHash_⬆) internal pure returns (bytes memory) {
380         (uint8 v, bytes32 r, bytes32 s) = vm.sign(signerPk_⬆, dataHash_⬆);
381         return abi.encodePacked(r, s, v);
382     }
383
        ftrace | funcSig
384     function _init_LiquidityPools() internal {
385         vm.startPrank(user1);
386         contract_Collateral.approve(address(contract_liquidityPool0), 200e18);
387         contract_liquidityPool0.depositAndLock(200e18, 0, 0);
388         console.log("    contract_liquidityPool0.totalSupply() -> %s", contract_liquidityPool0.totalSupply());
389         console.log(
390             "    contract_Collateral.balanceOf(address(contract_liquidityPool0)) -> %s",
391             contract_Collateral.balanceOf(address(contract_liquidityPool0))
392         );
393         vm.stopPrank();
394         vm.startPrank(user2);
395         contract_Collateral.approve(address(contract_liquidityPool1), 200e18);
396         contract_liquidityPool1.depositAndLock(200e18, 0, 0);
397         console.log("    contract_liquidityPool1.totalSupply() -> %s", contract_liquidityPool1.totalSupply());
398         console.log(
399             "    contract_Collateral.balanceOf(address(contract_liquidityPool1)) -> %s",
400             contract_Collateral.balanceOf(address(contract_liquidityPool1))
401         );
402         vm.stopPrank();
403     }
404 }
405
```

Results:

```
Traces:
  [568079] FullTestClosingFees::testFail_RemoveMargin()
    ├─ [0] VM::startPrank(0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC)
    │   └─ ← ()
    ├─ [1696] TradeManagerOrders::hashRemoveMarginFromPositionOrder(((0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856, 0, 115792089237316195423570985008687907853269984665640564039456584007913129639935), (1681307303, 999000000000000000, 1001000000000000000)), 0x398944ad0a014fc282d15770504a88d8f0dff8f097aa122d3fe54f07981e7eac, 0)) [staticcall]
    │   └─ ← 0x2adb472810f032e925c1835a1da66638297f856c9b9db6d1a3b4554ef55ff28e
    ├─ [0] VM::sign(<pk>, 0x2adb472810f032e925c1835a1da66638297f856c9b9db6d1a3b4554ef55ff28e) [staticcall]
    │   └─ ← 28, 0xe882a7551daf670545442adb185f6c571a01cbb9c37e7d530857f00b95da0f25, 0x7dfa67cf564459cb7a5bb441613f2bc1a38f89deb336777eaac5fc1372e6a701
    ├─ [0] VM::stopPrank()
    │   └─ ← ()
    ├─ [0] VM::warp(1681306703)
    │   └─ ← ()
    ├─ [0] VM::roll(17032047)
    │   └─ ← ()
    ├─ [21174] UserManager::setFeeSizes([0, 1, 2, 3, 4, 5, 6], [100, 100, 100, 100, 100, 100, 100])
    │   ├─ emit FeeSizeUpdated(feeIndex: 0, feeSize: 100)
    │   ├─ emit FeeSizeUpdated(feeIndex: 1, feeSize: 100)
    │   ├─ emit FeeSizeUpdated(feeIndex: 2, feeSize: 100)
    │   ├─ emit FeeSizeUpdated(feeIndex: 3, feeSize: 100)
    │   ├─ emit FeeSizeUpdated(feeIndex: 4, feeSize: 100)
    │   ├─ emit FeeSizeUpdated(feeIndex: 5, feeSize: 100)
    │   ├─ emit FeeSizeUpdated(feeIndex: 6, feeSize: 100)
    │   └─ ← ()
    ├─ [0] VM::prank(0xa0Ee7A142d267C1f36714E4a8F75612F20a79720)
    │   └─ ← ()
    ├─ [518500] TradeManagerOrders::removeMarginFromPositionViaSignature(((0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856, 0, 115792089237316195423570985008687907853269984665640564039456584007913129639935), (1681307303, 999000000000000000, 1001000000000000000)), 0x398944ad0a014fc282d15770504a88d8f0dff8f097aa122d3fe54f07981e7eac, 0), [], 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC, 0xe882a7551daf670545442adb185f6c571a01cbb9c37e7d530857f00b95da0f257dfa67cf564459cb7a5bb441613f2bc1a38f89deb336777eaac5fc1372e6a7011c)
    │   ├─ [2648] Controller::isOrderExecutor(0xa0Ee7A142d267C1f36714E4a8F75612F20a79720) [staticcall]
    │   │   └─ ← true
    │   ├─ [2673] Controller::checkTradePairActive(TradePair: [0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856]) [staticcall]
    │   │   └─ ← ()
    │   ├─ [3000] PRECOMPILE::ecrecover(0x2adb472810f032e925c1835a1da66638297f856c9b9db6d1a3b4554ef55ff28e, 28, 105167425876100585627887350739099194176511650049521648157406465160663277069 17, 56981534308770549113022072372796048812466229920137846284571541094074313806003 3) [staticcall]
    │   │   └─ ← 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC
```

```
    │   ├─ [2563] TradePair::positionIsShort(0) [staticcall]
    │   │   └─ ← false
    │   ├─ [10683] TradePair::getCurrentPrices() [staticcall]
    │   │   ├─ [2384] MockPriceFeedAdapter::markPriceMin() [staticcall]
    │   │   │   └─ ← 999900000000000000
    │   │   ├─ [2318] MockPriceFeedAdapter::markPriceMax() [staticcall]
    │   │   │   └─ ← 1010000000000000000
    │   │   └─ ← 999900000000000000, 1010000000000000000
    │   ├─ [2426] TradePair::collateral() [staticcall]
    │   │   └─ ← 0x6B175474E89094C44Da98b954EedeAC495271d0F
    │   ├─ [2580] Controller::orderRewardOfCollateral(0x6B175474E89094C44Da98b954EedeAC495271d0F) [staticcall]
    │   │   └─ ← 3000000000000000000
    │   ├─ [19175] 0x6B175474E89094C44Da98b954EedeAC495271d0F::transferFrom(0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC, 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720, 300000000000000000)
    │   │   ├─ emit Transfer(from: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC, to: 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720, value: 300000000000000000)
    │   │   └─ ← true
    │   ├─ emit OrderRewardTransfered(collateral: 0x6B175474E89094C44Da98b954EedeAC495271d0F, from: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC, to: 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720, orderReward: 300000000000000000)
    │   ├─ [350526] TradePair::removeMarginFromPosition(0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC, 0, 115792089237316195423570985008687907853269984665640564039456584007913129639935)
    │   │   ├─ [4912] FeeIntegralLib::8ca42358(0000000000000000000000000000000000000000000000000000000000000014) [delegatecall]
    │   │   │   └─ ← 0x0000000000000000000000000000000000000000000000000000011670dc155
    │   │   ├─ [42367] FeeIntegralLib::79b7e9f8(0000000000000000000000000000000000000000000000000000000000000014000000000000000000000000000000000000000000000000000000000000006b21ea85cba96beae6000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000) [delegatecall]
    │   │   │   ├─ [572] FundingFee::getFundingFeeRates(19762455528744491215590, 0, 2000000000000000, 3000000000000000) [delegatecall]
    │   │   │   │   └─ ← 3000000000000, 0
    │   │   │   └─ ← ()
    │   │   ├─ emit UpdatedFeeIntegrals(borrowFeeIntegral: 467029639722222 1, longFundingFeeIntegral: 2500000000000, shortFundingFeeIntegral: 0)
    │   │   ├─ [25020] FeeBufferLib::fa0cc746(0000000000000000000000000000000000000000000000000000000000000001b0000000000000000000000000000000000000000000000000017155ee7b728cbba) [delegatecall]
    │   │   │   └─ ← 0x0000000000000000000000000000000000000000000000000001150072dc95e98cc
    │   │   ├─ [2677] 0x6B175474E89094C44Da98b954EedeAC495271d0F::allowance(TradePair: [0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856], FeeManager: [0xf289aB36746386dF8084Af5B59bFE078A918c241]) [staticcall]
    │   │   │   └─ ← 0
    │   │   ├─ [677] 0x6B175474E89094C44Da98b954EedeAC495271d0F::allowance(TradePair: [0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856], FeeManager: [0xf289aB36746386dF8084Af5B59bFE078A918c241]) [staticcall]
    │   │   │   └─ ← 0
    │   │   ├─ [22414] 0x6B175474E89094C44Da98b954EedeAC495271d0F::approve(FeeManager: [0xf289aB36746386dF8084Af5B59bFE078A918c241], 124750499001497006 0)
    │   │   │   ├─ emit Approval(owner: TradePair: [0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856], spender: FeeManager: [0xf289aB36746386dF8084Af5B59bFE078A918c241], value: 124750499001497006 0)
    │   │   │   └─ ← true
    │   │   ├─ [171325] FeeManager::depositBorrowFees(0x6B175474E89094C44Da98b954EedeAC495271d0F, 124750499001497006 0)
    │   │   │   ├─ [562] Controller::isTradePair(TradePair: [0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856]) [staticcall]
    │   │   │   │   └─ ← true
    │   │   │   ├─ [2470] TradePair::liquidityPoolAdapter() [staticcall]
    │   │   │   │   └─ ← LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23]
    │   │   │   ├─ [25180] 0x6B175474E89094C44Da98b954EedeAC495271d0F::transferFrom(TradePair: [0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856], LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], 124750499001497006 0)
    │   │   │   │   ├─ emit Transfer(from: TradePair: [0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856], to: LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], value: 124750499001497006 0)
    │   │   │   │   └─ ← true
```

```
          ├─ [139132] LiquidityPoolAdapter::depositFees(1247504990014970060)
            ├─ [5648] LiquidityPool::availableLiquidity() [staticcall]
              ├─ [2602] 0x6B175474E89094C44Da98b954EedeAC495271d0F::balanceOf(LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA8359f]) [staticcall]
                └─ ← 200598802395209580839
              └─ ← 200598802395209580839
            ├─ [5648] LiquidityPool::availableLiquidity() [staticcall]
              ├─ [2602] 0x6B175474E89094C44Da98b954EedeAC495271d0F::balanceOf(LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475fFb7]) [staticcall]
                └─ ← 200598802395209580839
              └─ ← 200598802395209580839
            ├─ [2677] 0x6B175474E89094C44Da98b954EedeAC495271d0F::allowance(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA8359f]) [
staticcall]
              └─ ← 0
            ├─ [677] 0x6B175474E89094C44Da98b954EedeAC495271d0F::allowance(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA8359f]) [s
taticcall]
              └─ ← 0
            ├─ [22414] 0x6B175474E89094C44Da98b954EedeAC495271d0F::approve(LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA8359f], 623752495007485030)
              ├─ emit Approval(owner: LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], spender: LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA8359f], value: 623752495007485030)
              └─ ← true
            ├─ [36320] LiquidityPool::depositFees(623752495007485030)
              ├─ [2537] Controller::isLiquidityPoolAdapter(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23]) [staticcall]
                └─ ← true
              ├─ [7575] 0x6B175474E89094C44Da98b954EedeAC495271d0F::transferFrom(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA83
59f], 311876247503742515)
                ├─ emit Transfer(from: LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], to: LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA8359f], value: 311876247503742515)
                └─ ← true
              ├─ [602] 0x6B175474E89094C44Da98b954EedeAC495271d0F::balanceOf(LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA8359f]) [staticcall]
                └─ ← 200910678642713323354
              ├─ [3820] 0x6B175474E89094C44Da98b954EedeAC495271d0F::transferFrom(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA83
59f], 311876247503742515)
                ├─ emit Transfer(from: LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], to: LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA8359f], value: 311876247503742515)
                └─ ← true
              ├─ emit Transfer(from: 0x0000000000000000000000000000000000000000, to: LiquidityPool: [0x85a6d41Dd15aFf5d96519dCC6C4D81A4DEA8359f], value: 310462589256754489085460)
              ├─ emit DepositedFees(liquidityPoolAdapter: LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], amount: 623752495007485030)
              └─ ← ()
            ├─ [2677] 0x6B175474E89094C44Da98b954EedeAC495271d0F::allowance(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475fFb7]) [
staticcall]
              └─ ← 0
            ├─ [677] 0x6B175474E89094C44Da98b954EedeAC495271d0F::allowance(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475fFb7]) [s
taticcall]
              └─ ← 0
            ├─ [22414] 0x6B175474E89094C44Da98b954EedeAC495271d0F::approve(LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475fFb7], 623752495007485030)
              ├─ emit Approval(owner: LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], spender: LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475fFb7], value: 623752495007485030)
              └─ ← true
            ├─ [34720] LiquidityPool::depositFees(623752495007485030)
              ├─ [537] Controller::isLiquidityPoolAdapter(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23]) [staticcall]
                └─ ← true
              ├─ [7575] 0x6B175474E89094C44Da98b954EedeAC495271d0F::transferFrom(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475f
Fb7], 311876247503742515)
```

```
            ├─ [34720] LiquidityPool::depositFees(623752495007485030)
              ├─ [537] Controller::isLiquidityPoolAdapter(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23]) [staticcall]
                └─ ← true
              ├─ [7575] 0x6B175474E89094C44Da98b954EedeAC495271d0F::transferFrom(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475f
Fb7], 311876247503742515)
                ├─ emit Transfer(from: LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], to: LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475fFb7], value: 311876247503742515)
                └─ ← true
              ├─ [602] 0x6B175474E89094C44Da98b954EedeAC495271d0F::balanceOf(LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475fFb7]) [staticcall]
                └─ ← 200910678642713323354
              ├─ [3820] 0x6B175474E89094C44Da98b954EedeAC495271d0F::transferFrom(LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475f
Fb7], 311876247503742515)
                ├─ emit Transfer(from: LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], to: LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475fFb7], value: 311876247503742515)
                └─ ← true
              ├─ emit Transfer(from: 0x0000000000000000000000000000000000000000, to: LiquidityPool: [0xB6B59c61A1ACBbD3B07b62Ed573d93b03475fFb7], value: 310462589256754489085460)
              ├─ emit DepositedFees(liquidityPoolAdapter: LiquidityPoolAdapter: [0xda9D672b77fD718Ac9B80796E10263BB51080c23], amount: 623752495007485030)
              └─ ← ()
            ├─ emit DepositedProfit(tradePair: FeeManager: [0xf289aB36746386dF8084Af5B59bFE078A918c241], profit: 1247504990014970060)
            └─ ← ()
          └─ ← ()
        ├─ [2841] FeeIntegralLib::2d1aca7b(000000000000000000000000000000000000000000000000000000000000001400000000000000000000000000000000000000000000006b21ea85cba96beae6000000000000000000000000000000000000000
0000000000000000000000000) [delegatecall]
          ├─ [572] FundingFee::getFundingFeeRates(1976245528744491215590, 0, 200000000000000, 300000000000) [delegatecall]
            └─ ← 300000000000, 0
          └─ ← 0x0000000000000000000000000000000000000000000000000000003a352944000000000000000000000000000000000000000000000000000000000000000000
        ├─ [1158] FeeIntegralLib::96202e0e(0000000000000000000000000000000000000000000000000000000000000014) [delegatecall]
          └─ ← 0x00000000000000000000000000000000000000000000000000a5ec192985ad0d
        ├─ [78017] PositionMaths::2219a166(13da86008ba1c6922daee3e07db95305ef49ebced9f5467a0b8613fcc6b343e300000000000000000000000000000000000000000000000000a5ec192985ad0d000000000000000000000000000000000000000
0000000000000003a35294400) [delegatecall]
          └─ ← ()
        ├─ [2580] PositionMaths::4ad00c97(13da86008ba1c6922daee3e07db95305ef49ebced9f5467a0b8613fcc6b343e3ffffffffffffffffffffffffffffffffffffffffffffffffff21f494c589bffff) [delegatecall]
          └─ ← 0x08c379a000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000046506f736974696f6e4d617468733a3a5f72656d6f76654d617267696
e3a2063616e6e6f742072656d6f7665206d6f7265206d617267696e207468616e20617661696c61626c650000000000000000000000000000000000000000000000000000
        └─ ← "PositionMaths::_removeMargin: cannot remove more margin than available"
      └─ ← "PositionMaths::_removeMargin: cannot remove more margin than available"
    └─ ← "PositionMaths::_removeMargin: cannot remove more margin than available"
```

```
    ├─ [59398] TradeManagerOrders::closePositionViaSignature(((0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856, 115792089237316195423570985008687907853269984665640564039457584007913129639935), (1681307303, 9990000000000
00000, 1001000000000000000), 0x398944ad0a014fc282d15770504a88d8f0dff8f097aa122d3fe54f07981e7eac, 0), [], FakeSigner: [0x6c314c3B9Be6c5D46069Fa5a0610c49f07B84a01], 0x)
      ├─ [648] Controller::isOrderExecutor(0xa0Ee7A142d267C1f36714E4a8F75612F20a79720) [staticcall]
        └─ ← true
      ├─ [673] Controller::checkTradePairActive(TradePair: [0xfD9690e7Bbde3544AE4Fc9dae212933B95BCB856]) [staticcall]
        └─ ← ()
      ├─ [613] FakeSigner::isValidSignature(0x881c6814f477583650644188f3dd0af9f331f5ada7ecb5f855f0894c7a0f3e79, 0x) [staticcall]
        └─ ← 0x1626ba7e
      ├─ [563] TradePair::positionIsShort(115792089237316195423570985008687907853269984665640564039457584007913129639935) [staticcall]
        └─ ← false
      ├─ [2183] TradePair::getCurrentPrices() [staticcall]
        ├─ [384] MockPriceFeedAdapter::markPriceMin() [staticcall]
          └─ ← 999000000000000000
        ├─ [318] MockPriceFeedAdapter::markPriceMax() [staticcall]
          └─ ← 1010000000000000000
        └─ ← 999000000000000000, 1010000000000000000
      ├─ [426] TradePair::collateral() [staticcall]
        └─ ← 0x6B175474E89094C44Da98b954EedeAC495271d0F
      ├─ [580] Controller::orderRewardOfCollateral(0x6B175474E89094C44Da98b954EedeAC495271d0F) [staticcall]
        └─ ← 3000000000000000000
      ├─ [14375] 0x6B175474E89094C44Da98b954EedeAC495271d0F::transferFrom(FakeSigner: [0x6c314c3B9Be6c5D46069Fa5a0610c49f07B84a01], 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720, 3000000000000000000)
        ├─ emit Transfer(from: FakeSigner: [0x6c314c3B9Be6c5D46069Fa5a0610c49f07B84a01], to: 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720, value: 3000000000000000000)
        └─ ← true
      ├─ emit OrderRewardTransfered(collateral: 0x6B175474E89094C44Da98b954EedeAC495271d0F, from: FakeSigner: [0x6c314c3B9Be6c5D46069Fa5a0610c49f07B84a01], to: 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720, orderRew
ard: 3000000000000000000)
      ├─ [806] TradePair::closePosition(FakeSigner: [0x6c314c3B9Be6c5D46069Fa5a0610c49f07B84a01], 0)
        └─ ← "TradePair::_verifyOwner: not the owner"
      └─ ← "TradePair::_verifyOwner: not the owner"
    └─ ← "TradePair::_verifyOwner: not the owner"
```

MANUAL TESTING

THANK YOU FOR CHOOSING

// HALBORN