

# **HMX Audit Report**

Jun 27, 2023





# **Table of Contents**

Summary	2
Overview	3
Issues	4
[WP-H1] _executionFee can be set to an arbitrary value when _shouldWrap = false in createAddLiquidityOrder().	4
[WP-M2] Malimplementation of getFundingFee()	8
[WP-L4] Failed native token transfers should be handled properly.	13
[WP-L5] LiquidityHandler.executeOrder() maxExecutionChuck feature has an inaccurate implementation.	17
[WP-L6] convertSGlpCollateral() can lower the account's equity, therefore it should check the IMR	21
[WP-L7] ConvertedGlpStrategy Lack of slippage control	23
[WP-L8] Wrong implementation of array length validation	26
[WP-G9] EcoPyth.buildPriceUpdateData() Optimizations	30
[WP-M10] LiquidityHandler.cancelLiquidityOrder()' should refunds the execution fee	32
[WP-M11] Cancelled WithdrawOrder should be skipped	36
[WP-G12] Redundant if (uint8Value > 255) check code	42
[WP-G13] proportionalElapsedInDay() can be simplified.	43
<pre>[WP-G14] TradeService.increasePosition() redundant HMXLib.abs(_vars.position.positionSizeE30) &gt; 0 check.</pre>	44
[WP-G15] Unnecessary sload	45
[WP-M16] User's balance in tradingStaking may not be decreased as expected due to precision loss.	47
[WP-M17] Malicious users can prevent the execution of executeLiquidity() by transferring 1 wei token to the vaultStorage contract.	51

	HMX
[WP-N19] Dev related codes to be removed	56
Appendix	57
Disclaimer	58



# Summary

This report has been prepared for HMX smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.



# Overview

# **Project Summary**

Project Name	HMX
Codebase	https://github.com/eqbtech/HMX-contracts
Commit	083c5986b558335ec576841d7c4f050a1bcd33b4
Language	Solidity

# **Audit Summary**

Delivery Date	Jun 27, 2023
Audit Methodology	Static Analysis, Manual Review
Total Isssues	17



# [WP-H1] \_executionFee can be set to an arbitrary value when \_shouldWrap = false in createAddLiquidityOrder().

High

### **Issue Description**

There is no check in <code>createAddLiquidityOrder()</code> to ensure that the caller has paid for the <code>\_executionFee</code> specified in the calldata when <code>\_shouldWrap</code> is set to <code>false</code>. This allows an attacker to specify an extremely high <code>executionFee</code>, potentially all the balance of the <code>LiquidityHandler</code> contract.

Although the current implementation of <code>cancelLiquidityOrder()</code> does not actually refund the execution fee as stated in the comment, an attacker cannot exploit this by simply calling <code>createAddLiquidityOrder()</code> and <code>cancelLiquidityOrder()</code>, the issue can still result in paying the wrong executionFee to feeReceiver.

https://github.com/perp88/v2-evm/blob/95ea5a58c319139f4e4c68790dfa54dec3ffb808/src/handlers/LiquidityHandler.sol#L391-L393

```
/// @notice Cancels the specified add/remove liquidity order and refunds the
execution fee.

/// @param _orderIndex Index of the order to cancel.

function cancelLiquidityOrder(uint256 _orderIndex) external nonReentrant {
```

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/handlers/LiquidityHandler.sol#L146-L198

```
146
     function createAddLiquidityOrder(
147
         address _tokenIn,
148
         uint256 _amountIn,
149
         uint256 _minOut,
         uint256 _executionFee,
150
151
         bool _shouldWrap
        ) external payable nonReentrant onlyAcceptedToken( tokenIn) returns (uint256
152
     _orderId) {
153
         // pre validate
154
         LiquidityService(liquidityService).validatePreAddRemoveLiquidity(_amountIn);
```



```
155
          if (_executionFee < minExecutionOrderFee) revert</pre>
     ILiquidityHandler_InsufficientExecutionFee();
156
          if (_shouldWrap && _tokenIn !=
     ConfigStorage(LiquidityService(liquidityService).configStorage()).weth())
157
            revert ILiquidityHandler_NotWNativeToken();
158
159
         if ( shouldWrap) {
            if (msg.value != _amountIn + _executionFee) revert
160
     ILiquidityHandler InCorrectValueTransfer();
161
          } else {
162
            if (msg.value != minExecutionOrderFee) revert
     ILiquidityHandler_InCorrectValueTransfer();
163
            IERC20Upgradeable(_tokenIn).safeTransferFrom(msg.sender, address(this),
     _amountIn);
164
         }
165
         // convert native to WNative (including executionFee)
166
         _transferInETH();
167
168
169
         _orderId = liquidityOrders.length;
170
171
         liquidityOrders.push(
172
            LiquidityOrder({
173
              account: payable(msg.sender),
174
              orderId: _orderId,
175
              token: tokenIn,
176
              amount: _amountIn,
              minOut: _minOut,
177
              actualAmountOut: 0,
178
              isAdd: true,
179
              executionFee: _executionFee,
180
181
              isNativeOut: _shouldWrap,
              createdTimestamp: uint48(block.timestamp),
182
              executedTimestamp: 0,
183
              status: LiquidityOrderStatus.PENDING
184
           })
185
186
         );
187
         emit LogCreateAddLiquidityOrder(
188
     @@ 189,195 @@
196
          );
197
          return _orderId;
```



```
198 }
```

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/handlers/LiquidityHandler.sol#L270-L349

```
function executeOrder(
270
271
         uint256 _endIndex,
          address payable _feeReceiver,
272
273
         bytes32[] calldata priceData,
         bytes32[] calldata _publishTimeData,
274
         uint256 _minPublishTime,
275
         bytes32 _encodedVaas
276
        ) external nonReentrant onlyOrderExecutor {
277
     @@ 278,302 @@
303
         for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {</pre>
304
            _order = liquidityOrders[i];
305
            if (_order.amount > 0) {
306
              _executionFee = _order.executionFee;
307
308
309
              try this.executeLiquidity(_order) returns (uint256 actualOut) {
     @@ 310,329 @@
330
             // assign exec time
331
332
              _order.executedTimestamp = uint48(block.timestamp);
333
              _totalFeeReceiver += _executionFee;
334
335
             // save to executed order first
              accountExecutedLiquidityOrders[_order.account].push(_order);
336
              // clear executed liquidity order
337
338
              delete liquidityOrders[i];
339
            }
340
341
            unchecked {
342
              ++i;
343
            }
344
         }
345
346
          nextExecutionOrderIndex = _endIndex + 1;
```



```
// Pay total collected fees to the executor

transferOutETH(_totalFeeReceiver, _feeReceiver);

}
```





# [WP-M2] Malimplementation of getFundingFee()

### Medium

### **Issue Description**

The comment on line 1023 of Calculator.sol states that if the returned fundingFee is less than 0, it means the trader received the fee.

However, in the current implementation, the user can not receive the funding fee correctly.

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/contracts/Calculator.sol#L1002-L1030

```
function getFundingFee(
1002
          bool isLong,
1003
          uint256 _size,
1004
1005
          int256 _currentFundingAccrued,
          int256 lastFundingAccrued
1006
        ) public pure returns (int256 fundingFee) {
1007
          if (_size == 0) return 0;
1008
          int256 _fundingAccrued = _currentFundingAccrued - _lastFundingAccrued;
1009
1010
1011
          // IF _fundingAccrued < 0, LONG positions pay fees to SHORT and SHORT
      positions receive fees from LONG
          // IF _fundingAccrued > 0, LONG positions receive fees from SHORT and SHORT
1012
      pay fees to LONG
          fundingFee = (int256(_size) * _fundingAccrued) / int64(RATE_PRECISION);
1013
1014
1015
          // Position Exposure | Funding Rate | Fund Flow
1016
          // (isLong)
                               | (fundingRate > 0) | (traderMustPay)
1017
1018
          // true
                               | true
                                                   | false (fee reserve -> trader)
                               | false
1019
          // true
                                                   | true (trader -> fee reserve)
         // false
                               | true
                                                   | true (trader -> fee reserve)
1020
          // false
                                | false
                                                   | false (fee reserve -> trader)
1021
1022
          // If fundingFee is negative mean Trader receives Fee
1023
1024
          // If fundingFee is positive mean Trader pays Fee
          if (_isLong) {
1025
1026
            return -fundingFee;
```



Assuming the user holds a long position ( \_isLong == true ) and should receive a fee (returned fundingFee < 0 ), TradeHelper.sol at L417 will invert the fundingFee to a positive number and enter the \_updateAccumFundingLong() function.

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/helpers/TradeHelper.sol#L379-L422

```
function updateFeeStates(
379
         bytes32 _positionId,
380
         address subAccount,
381
382
         PerpStorage.Position memory _position,
         uint256 sizeDelta,
383
384
         uint32 _positionFeeBPS,
385
         uint8 assetClassIndex,
386
         uint256 _marketIndex
387
        ) internal returns (uint256 _tradingFee, uint256 _borrowingFee, int256
     fundingFee) {
         // SLOAD
388
         Calculator _ calculator = calculator;
389
390
         // Calculate the trading fee
391
         _tradingFee = (_sizeDelta * _positionFeeBPS) / BPS;
392
         emit LogSettleTradingFeeValue(_positionId, _subAccount, _tradingFee);
393
394
         // Calculate the borrowing fee
395
         _borrowingFee = _calculator.getBorrowingFee(
396
397
           _assetClassIndex,
           _position.reserveValueE30,
398
399
           _position.entryBorrowingRate
400
         );
         // Update global state
401
         _accumSettledBorrowingFee(_assetClassIndex, _borrowingFee);
402
         emit LogSettleBorrowingFeeValue(_positionId, _subAccount, _borrowingFee);
403
404
         // Calculate the funding fee
405
```



```
406
         // We are assuming that the market state has been updated with the latest
     funding rate
407
         bool _isLong = _position.positionSizeE30 > 0;
408
         fundingFee = calculator.getFundingFee(
409
           _isLong,
410
           HMXLib.abs(_position.positionSizeE30),
           PerpStorage(perpStorage).getMarketByIndex( marketIndex).fundingAccrued,
411
           _position.lastFundingAccrued
412
413
         );
414
         // Update global state
415
416
         _isLong
            ? updateAccumFundingLong( marketIndex, - fundingFee)
417
418
            : _updateAccumFundingShort(_marketIndex, -_fundingFee);
         emit LogSettleFundingFeeValue(_positionId, _subAccount, uint256(_fundingFee));
419
420
         return ( tradingFee, borrowingFee, fundingFee);
421
422
```

In the \_updateAccumFundingLong function, the inverted fundingFee (>0) will be added to \_market.accumFundingLong , making \_market.accumFundingLong positive.

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/helpers/TradeHelper.sol#L837-L851

```
function _updateAccumFundingLong(uint256 _marketIndex, int256 fundingLong)
     internal {
         PerpStorage _perpStorage = PerpStorage(perpStorage);
838
839
         PerpStorage.Market memory market =
     _perpStorage.getMarketByIndex(_marketIndex);
840
          _market.accumFundingLong += fundingLong;
841
842
         _perpStorage.updateMarket(_marketIndex, _market);
843
       }
844
       function _updateAccumFundingShort(uint256 _marketIndex, int256 fundingShort)
845
     internal {
846
         PerpStorage perpStorage = PerpStorage(perpStorage);
         PerpStorage.Market memory market =
847
     _perpStorage.getMarketByIndex(_marketIndex);
848
```



```
__market.accumFundingShort += fundingShort;

850     __perpStorage.updateMarket(_marketIndex, __market);

851 }
```

In CrossMarginService, fundingFeeBookValue is the money that should be paid to the user. However, since accumFundingLong is positive, it will not actually be added to fundingFeeBookValue, causing the user to not receive the money. This contradicts the assumption that the user holds a long position and should receive the fee.

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/services/CrossMarginService.sol#L209-L261

```
function withdrawFundingFeeSurplus(address stableToken) external nonReentrant
209
     onlyWhitelistedExecutor {
210
         // SLOAD
211
         ConfigStorage _configStorage = ConfigStorage(configStorage);
212
          PerpStorage perpStorage = PerpStorage(perpStorage);
213
         VaultStorage vaultStorage = VaultStorage(vaultStorage);
214
          OracleMiddleware _oracle = OracleMiddleware(_configStorage.oracle());
215
216
         WithdrawFundingFeeSurplusVars memory vars;
217
218
         // Get funding Fee LONG & SHORT on each market to find positive values
         // positive value mean how much protocol book funding fee value that will be
219
     paid to trader
220
         // Loop through all markets to sum funding fee on LONG and SHORT sides
          uint256 len = configStorage.getMarketConfigsLength();
221
         for (uint256 i = 0; i < len; ) {</pre>
222
223
            PerpStorage.Market memory _market = _perpStorage.getMarketByIndex(i);
224
225
            if ( market.accumFundingLong < 0) vars.fundingFeeBookValue +=</pre>
     uint256(-_market.accumFundingLong);
            if (_market.accumFundingShort < 0) _vars.fundingFeeBookValue +=</pre>
226
     uint256(-_market.accumFundingShort);
227
228
            unchecked {
229
              ++i;
230
            }
231
          }
232
```



```
233
         // Calculate value of current Funding fee reserve
234
         _vars.tokenAssetId = _configStorage.tokenAssetIds(_stableToken);
235
         _vars.tokenDecimal = _configStorage.getAssetTokenDecimal(_stableToken);
         (_vars.tokenPrice, ) = _oracle.getLatestPrice( vars.tokenAssetId, false);
236
237
         _vars.fundingFeeAmount = _vaultStorage.fundingFeeReserve(_stableToken);
         _vars.totalFundingFeeReserveValueE30 = (_vars.fundingFeeAmount *
238
     vars.tokenPrice) / (10 ** vars.tokenDecimal);
239
         // If fundingFeeBookValue > totalFundingFeeReserveValueE30 means protocol has
240
     exceed balance of fee reserved for paying to traders
         // Funding fee surplus = totalFundingFeeReserveValueE30 - fundingFeeBookValue
241
         if ( vars.fundingFeeBookValue > vars.totalFundingFeeReserveValueE30 ||
242
     ( vars.totalFundingFeeReserveValueE30 == 0))
           revert ICrossMarginHandler_NoFundingFeeSurplus();
243
244
         _vars.fundingFeeSurplusValue = _vars.totalFundingFeeReserveValueE30 -
245
     vars.fundingFeeBookValue;
         // Transfer surplus amount to HLP
246
247
         {
            (uint256 _repayAmount, uint256 _repayValue) = _getRepayAmount(
248
249
             _configStorage,
250
             _oracle,
251
             _vars.fundingFeeAmount,
252
             vars.fundingFeeSurplusValue,
253
             stableToken
254
           );
255
           _vaultStorage.withdrawSurplusFromFundingFeeReserveToHLP(_stableToken,
256
     _repayAmount);
           vars.fundingFeeSurplusValue -= repayValue;
257
258
         }
259
         emit LogWithdrawFundingFeeSurplus( vars.fundingFeeSurplusValue);
260
261
```





# [WP-L4] Failed native token transfers should be handled properly.

Low

### **Issue Description**

The system frequently ignores the result of native token transfer calls.

Some do so intentionally:

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/account-abstraction/BaseAccount.sol#L104-L110

```
function _payPrefund(uint256 missingAccountFunds) internal virtual {
   if (missingAccountFunds != 0) {
      (bool success, ) = payable(msg.sender).call{ value: missingAccountFunds,
      gas: type(uint256).max }("");

(success);

//ignore failure (its EntryPoint's job to verify, not account.)

}

100 }
```

Others are speculated to do it in order to prevent the compiler from generating warnings about unused variables, based on the comment "// shhh compiler":

https://github.com/perp88/v2-evm/blob/2ed66c3b30edf7f56fcd97f437f148d66af3bc3a/src/handlers/LiquidityHandler.sol#L462-L472

```
462
     function _transferOutETH(uint256 _amountOut, address _receiver) private {
463
     IWNative(ConfigStorage(LiquidityService(liquidityService).configStorage()).weth()).withdraw(_
464
         // slither-disable-next-line arbitrary-send-eth
465
         // To mitigate potential attacks, the call method is utilized,
         // allowing the contract to bypass any revert calls from the destination
466
     address.
         // By setting the gas limit to 2300, equivalent to the gas limit of the
467
     transfer method,
468
         // the transaction maintains a secure execution."
```



```
(bool success, ) = _receiver.call{ value: _amountOut, gas: 2300 }("");
// shhh compiler
success;
472 }
```

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/handlers/LimitTradeHandler.sol#L940-L950

```
940
       function transferOutETH(uint256 amountOut, address receiver) private {
         IWNative(weth).withdraw( amountOut);
941
         // slither-disable-next-line arbitrary-send-eth
942
         // To mitigate potential attacks, the call method is utilized,
943
         // allowing the contract to bypass any revert calls from the destination
944
     address.
         // By setting the gas limit to 2300, equivalent to the gas limit of the
945
     transfer method,
         // the transaction maintains a secure execution."
946
         (bool success, ) = receiver.call{ value: amountOut, gas: 2300 }("");
947
         // shhh compiler
948
949
         success;
950
```

However, we think that the native token transfer calls in LiquidityHandler.sol#\_transferOutETH() and LimitTradeHandler.sol#\_transferOutETH() should be handled more appropriately because the lack of require(success, "...") will cause L382 \_order.account to not revert even if it did not receive the money (reducing Liquidity but not receiving money).

https://github.com/perp88/v2-evm/blob/a6eddbd6a6378da3e9dc6f6f4d10b721fa67ba7e/src/handlers/LiquidityHandler.sol#L359-L389

```
function executeLiquidity(LiquidityOrder calldata _order) external returns
  (uint256 _amountOut) {

// if not in executing state, then revert

if (msg.sender != address(this)) revert ILiquidityHandler_Unauthorized();

if (_order.isAdd) {
```



```
@@ 364,372 @@
373
          } else {
374
            amountOut = LiquidityService(liquidityService).removeLiquidity(
              _order.account,
375
376
              _order.token,
377
              _order.amount,
              _order.minOut
378
379
            );
380
            if (_order.isNativeOut) {
381
382
              _transferOutETH(_amountOut, payable(_order.account));
383
            } else {
384
              IERC20Upgradeable( order.token).safeTransfer( order.account, amountOut);
385
            }
386
387
            return _amountOut;
388
          }
389
        }
```

### Recommendation

Consider sending WNative instead when a native token transfer fails:

```
940
       function _transferOutETH(uint256 _amountOut, address _receiver) private {
941
         IWNative(weth).withdraw( amountOut);
942
         // slither-disable-next-line arbitrary-send-eth
943
         // To mitigate potential attacks, the call method is utilized,
944
         // allowing the contract to bypass any revert calls from the destination
     address.
945
         // By setting the gas limit to 2300, equivalent to the gas limit of the
     transfer method,
         // the transaction maintains a secure execution."
946
         (bool success, ) = receiver.call{ value: amountOut, gas: 2300 }("");
947
948
         // send WNative instead when native token transfer fail
949
         if (!success) {
             IWNative(weth).deposit(_amountOut);
950
             IWNative(weth).transfer(_receiver, _amountOut);
951
952
         }
```







# [WP-L5] LiquidityHandler.executeOrder() maxExecutionChuck feature has an inaccurate implementation.

Low

## **Issue Description**

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/handlers/LiquidityHandler.sol#L87

```
87 uint256 public maxExecutionChuck; // maximum execution order sizes per request
```

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/handlers/LiquidityHandler.sol#L270-L349

```
270
     function executeOrder(
         uint256 _endIndex,
271
         address payable _feeReceiver,
272
         bytes32[] calldata _priceData,
273
274
         bytes32[] calldata _publishTimeData,
275
         uint256 _minPublishTime,
         bytes32 encodedVaas
276
277
        ) external nonReentrant onlyOrderExecutor {
         uint256  nextExecutionOrderIndex = nextExecutionOrderIndex;
278
279
         // Get the number of liquidity orders
280
         uint256 _orderLength = liquidityOrders.length;
281
282
283
         // Ensure there are orders to execute
284
         if (_nextExecutionOrderIndex == _orderLength) revert
     ILiquidityHandler_NoOrder();
285
286
         // Set the end index to the latest order index if it exceeds the number of
     orders
         uint256 _latestOrderIndex = _orderLength - 1;
287
         if (_endIndex > _latestOrderIndex) {
288
           _endIndex = _latestOrderIndex;
289
         }
290
291
```



```
292
         // split execution into chunk for preventing exceed block gas limit
293
          if (_endIndex - _nextExecutionOrderIndex > maxExecutionChuck)
294
            _endIndex = _nextExecutionOrderIndex + maxExecutionChuck;
295
296
         // slither-disable-next-line arbitrary-send-eth
297
          IEcoPyth(pyth).updatePriceFeeds(_priceData, _publishTimeData, _minPublishTime,
     encodedVaas);
298
299
         // Initialize variables for the execution loop
          LiquidityOrder memory _order;
300
         uint256 _totalFeeReceiver;
301
302
         uint256 executionFee;
303
304
         for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {</pre>
     @@ 305,340 @@
341
            unchecked {
342
              ++i;
            }
343
344
         }
345
         nextExecutionOrderIndex = endIndex + 1;
346
         // Pay total collected fees to the executor
347
348
          _transferOutETH(_totalFeeReceiver, _feeReceiver);
349
```

1. The actual number of executed orders is <code>\_endIndex-\_nextExecutionOrderIndex+1</code> , because <code>LiquidityHandler.sol</code> L304 is a closed interval. This also causes <code>maxExecutionChuck</code> to constrain the execution quantity by 1 more than intended.

When L293-L294 is executed, \_endIndex-\_next = \_nextExecutionOrderIndex + maxExecutionChuck , causing L304 to execute maxExecutionChuck+1 orders.

2. Typo: maxExecutionChuck -> maxExecutionChunk

#### Given:

```
• nextExecutionOrderIndex : 1
```

•  $\max ExecutionChuck:10$ 

When: executeOrder({\_endIndex: 20, ...})

Then:



- L293 \_endIndex \_nextExecutionOrderIndex > maxExecutionChuck is true, since 20 1 > 10
- L294 \_endIndex = \_nextExecutionOrderIndex + maxExecutionChuck ( \_endIndex = 1 + 10 ) sets \_endIndex to 11
- L304 L344 will execute orders from \_nextExecutionOrderIndex (1) up to and including \_endIndex (11), for a total of 11 orders
- L346 updates the storage of nextExecutionOrderIndex to endIndex + 1 (12)

### Summary:

- Current implementation: when <u>\_endIndex</u> parameter is too large, it is automatically adjusted to execute only <u>\_maxExecutionChuck + 1</u> (11) orders
- Expected implementation: when \_endIndex parameter is too large, it should be automatically adjusted to execute only maxExecutionChuck (10) orders

### Recommendation

Consider changing to:

87 **uint256 public** maxExecutionChunk; // maximum execution order sizes per request

```
270
       function executeOrder(
         uint256 _endIndex,
271
         address payable _feeReceiver,
272
273
         bytes32[] calldata _priceData,
         bytes32[] calldata _publishTimeData,
274
275
         uint256 minPublishTime,
276
         bytes32 _encodedVaas
277
       ) external nonReentrant onlyOrderExecutor {
278
         uint256  nextExecutionOrderIndex = nextExecutionOrderIndex;
279
280
         // Get the number of liquidity orders
         uint256 _orderLength = liquidityOrders.length;
281
282
283
         // Ensure there are orders to execute
284
         if ( nextExecutionOrderIndex == orderLength) revert
     ILiquidityHandler NoOrder();
285
         // Set the end index to the latest order index if it exceeds the number of
286
     orders
```



```
287
          uint256 _latestOrderIndex = _orderLength - 1;
288
          if (_endIndex > _latestOrderIndex) {
289
           _endIndex = _latestOrderIndex;
290
         }
291
292
         // split execution into chunk for preventing exceed block gas limit
293
         if (_endIndex - _nextExecutionOrderIndex > maxExecutionChunk - 1)
294
           _endIndex = _nextExecutionOrderIndex + maxExecutionChunk - 1;
295
296
         // slither-disable-next-line arbitrary-send-eth
297
         IEcoPyth(pyth).updatePriceFeeds(_priceData, _publishTimeData, _minPublishTime,
     _encodedVaas);
298
         // Initialize variables for the execution loop
299
         LiquidityOrder memory _order;
300
         uint256 _totalFeeReceiver;
301
         uint256 executionFee;
302
303
304
         for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {</pre>
     @@ 305,340 @@
           unchecked {
341
342
              ++i;
343
           }
344
         }
345
         nextExecutionOrderIndex = endIndex + 1;
346
347
         // Pay total collected fees to the executor
348
         _transferOutETH(_totalFeeReceiver, _feeReceiver);
349
```





# [WP-L6] convertSGlpCollateral() can lower the account's equity, therefore it should check the IMR

Low

### **Issue Description**

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/services/CrossMarginService.sol#L263-L281

```
function convertSGlpCollateral(
263
264
         address _primaryAccount,
265
         uint8 subAccountId,
         address tokenOut,
266
         uint256 _amountIn
267
     ) external nonReentrant onlyWhitelistedExecutor returns (uint256 _amountOut) {
268
         // Get trader's sub-account address
269
270
         VaultStorage _vaultStorage = VaultStorage(vaultStorage);
         ConfigStorage configStorage = ConfigStorage(configStorage);
271
272
         _amountOut = ConvertedGlpStrategy(convertedSglpStrategy).execute(_tokenOut,
     _amountIn);
273
         // Adjusting trader balance
274
         address _subAccount = HMXLib.getSubAccount(_primaryAccount, _subAccountId);
275
         _vaultStorage.decreaseTraderBalance(_subAccount, _configStorage.sglp(),
276
     amountIn);
         vaultStorage.increaseTraderBalance( subAccount, tokenOut, amountOut);
277
278
279
         emit LogConvertSGlpCollateral(_primaryAccount, _subAccountId, _tokenOut,
     _amountIn, _amountOut);
280
         return _amountOut;
281
```

Because converting sglp to other tokens incurs a cost, convertSGlpCollateral() may lower the account's equity. However, there is no IMR check in convertSGlpCollateral(). For reference, withdrawCollateral() checks IMR after the withdrawal.

https://github.com/perp88/v2-evm/blob/95ea5a58c319139f4e4c68790dfa54dec3ffb808/src/services/CrossMarginService.sol#L172-L204



```
172
       function withdrawCollateral(
          address _primaryAccount,
173
174
         uint8 subAccountId,
175
          address token,
176
         uint256 _amount,
177
          address _receiver
        ) external nonReentrant onlyWhitelistedExecutor onlyAcceptedToken( token) {
178
179
         // SLOAD
180
          Calculator calculator = Calculator(calculator);
181
         VaultStorage _vaultStorage = VaultStorage(vaultStorage);
182
183
184
         // Get trader's sub-account address
         address _subAccount = HMXLib.getSubAccount(_primaryAccount, _subAccountId);
185
186
         // Get current collateral token balance of trader's account
187
         // and deduct with new token withdrawing amount
188
         uint256 _oldBalance = _vaultStorage.traderBalances(_subAccount, _token);
189
         if (_amount > _oldBalance) revert ICrossMarginService_InsufficientBalance();
190
191
192
         // Decrease collateral token balance
193
         _vaultStorage.decreaseTraderBalance(_subAccount, _token, _amount);
194
195
         // Calculate validation for if new Equity is below IMR or not
          int256 equity = _calculator.getEquity(_subAccount, 0, 0);
196
197
         if (equity < 0 | | uint256(equity) < _calculator.getIMR(_subAccount))</pre>
198
            revert ICrossMarginService WithdrawBalanceBelowIMR();
199
200
         // Transfer withdrawing token from VaultStorage to destination wallet
         _vaultStorage.pushToken(_token, _receiver, _amount);
201
202
203
          emit LogWithdrawCollateral(_primaryAccount, _subAccount, _token, _amount,
      _receiver);
204
```





# [WP-L7] ConvertedGlpStrategy Lack of slippage control

Low

## **Issue Description**

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/strategies/ConvertedGlpStrategy.sol#L54-L73

```
function execute(address _tokenOut, uint256 _amount) external onlyWhitelist
     returns (uint256 _amountOut) {
         // 1. Build calldata.
55
         bytes memory _callData = abi.encodeWithSelector(
56
           IGmxRewardRouterV2.unstakeAndRedeemGlp.selector,
57
           _tokenOut,
58
59
           _amount,
60
           0,
           address(this)
         );
62
63
64
         // 2. withdraw sglp from GMX
65
         bytes memory _cookResult = vaultStorage.cook(address(sglp),
     address(rewardRouter), _callData);
66
         _amountOut = abi.decode(_cookResult, (uint256));
67
         // 3. Transfer token to vaultStorage
69
         IERC20Upgradeable(_tokenOut).safeTransfer(address(vaultStorage), _amountOut);
         vaultStorage.pullToken(_tokenOut);
70
71
72
         return _amountOut;
73
       }
```

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/services/CrossMarginService.sol#L263-L281

```
function convertSGlpCollateral(

address _primaryAccount,

uint8 _subAccountId,

address _tokenOut,

uint256 _amountIn
```



```
268
      ) external nonReentrant onlyWhitelistedExecutor returns (uint256 _amountOut) {
269
         // Get trader's sub-account address
270
         VaultStorage vaultStorage = VaultStorage(vaultStorage);
271
         ConfigStorage configStorage = ConfigStorage(configStorage);
         _amountOut = ConvertedGlpStrategy(convertedSglpStrategy).execute(_tokenOut,
272
     _amountIn);
273
274
         // Adjusting trader balance
275
         address _subAccount = HMXLib.getSubAccount(_primaryAccount, _subAccountId);
276
         _vaultStorage.decreaseTraderBalance(_subAccount, _configStorage.sglp(),
     _amountIn);
         _vaultStorage.increaseTraderBalance(_subAccount, _tokenOut, _amountOut);
277
278
279
         emit LogConvertSGlpCollateral(_primaryAccount, _subAccountId, _tokenOut,
     _amountIn, _amountOut);
280
         return _amountOut;
281
```

https://github.com/gmx-io/gmx-contracts/blob/master/contracts/staking/RewardRouterV2.sol#LL159C5-L170C6

```
function unstakeAndRedeemGlp(address _tokenOut, uint256 _glpAmount, uint256
     _minOut, address _receiver) external nonReentrant returns (uint256) {
160
         require( glpAmount > 0, "RewardRouter: invalid glpAmount");
161
162
         address account = msg.sender;
163
         IRewardTracker(stakedGlpTracker).unstakeForAccount(account, feeGlpTracker,
     glpAmount, account);
164
         IRewardTracker(feeGlpTracker).unstakeForAccount(account, glp, glpAmount,
     account);
165
         uint256 amountOut = IGlpManager(glpManager).removeLiquidityForAccount(account,
     tokenOut, glpAmount, minOut, receiver);
166
167
         emit UnstakeGlp(account, _glpAmount);
168
169
         return amountOut;
170
     }
```

GMX's RewardRouterV2.sol#unstakeAndRedeemGlp() allows the caller to specify a \_minOut as the



slippage control. However, ConvertedGlpStrategy does not support slippage control.

This exposes the transaction to sandwich attacks or other kinds of MEV attacks.





# [WP-L8] Wrong implementation of array length validation

Low

## **Issue Description**

### BotHandler.sol#deleverages()

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/handlers/BotHandler.sol#L257-L281

```
257
     function deleverages(
258
          address[] memory _accounts,
259
          uint8[] memory _subAccountIds,
260
         uint256[] memory _marketIndexes,
         address[] memory _tpTokens,
261
         bytes32[] memory _priceData,
262
         bytes32[] memory _publishTimeData,
263
         uint256 minPublishTime,
264
         bytes32 _encodedVaas
265
266
        ) external payable nonReentrant onlyPositionManager {
         // pre-validation
267
         if (
268
269
           _accounts.length != _subAccountIds.length &&
           _subAccountIds.length != _marketIndexes.length &&
270
           _marketIndexes.length != _tpTokens.length
271
272
          ) revert IBotHandler_InvalidArray();
273
274
         // Feed Price
275
         // slither-disable-next-line arbitrary-send-eth
         IEcoPyth(pyth).updatePriceFeeds(_priceData, _publishTimeData, _minPublishTime,
276
     encodedVaas);
277
278
          _deleverages(_accounts, _subAccountIds, _marketIndexes, _tpTokens);
279
280
          emit LogDeleverages(_accounts, _subAccountIds, _marketIndexes, _tpTokens);
281
```

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/handlers/BotHandler.sol#L283-L299



```
283
      function _deleverages(
284
          address[] memory _accounts,
285
          uint8[] memory _subAccountIds,
286
          uint256[] memory marketIndexes,
          address[] memory _tpTokens
287
288
        ) internal nonReentrant {
          // SLOAD
289
290
          TradeService _tradeService = TradeService(tradeService);
291
          uint256 len = accounts.length;
292
          for (uint256 i; i < len; ) {</pre>
293
            _tradeService.validateDeleverage();
294
            _tradeService.forceClosePosition(_accounts[i], _subAccountIds[i],
      _marketIndexes[i], _tpTokens[i]);
295
            unchecked {
296
              ++i;
            }
297
298
          }
299
```

### 2. BotHandler.sol#closeDelistedMarketPositions()

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/handlers/BotHandler.sol#L330-L354

```
function closeDelistedMarketPositions(
330
331
          address[] calldata accounts,
332
         uint8[] calldata _subAccountIds,
         uint256[] calldata marketIndexes,
333
334
         address[] calldata _tpTokens,
335
         bytes32[] memory _priceData,
336
         bytes32[] memory _publishTimeData,
         uint256 _minPublishTime,
337
          bytes32 _encodedVaas
338
339
        ) external payable nonReentrant onlyPositionManager {
340
         // pre-validation
341
         if (
342
            _accounts.length != _subAccountIds.length &&
            _subAccountIds.length != _marketIndexes.length &&
343
            _marketIndexes.length != _tpTokens.length
344
          ) revert IBotHandler_InvalidArray();
345
346
```



```
347
         // Feed Price
348
         // slither-disable-next-line arbitrary-send-eth
349
         IEcoPyth(pyth).updatePriceFeeds(_priceData, _publishTimeData, _minPublishTime,
     encodedVaas);
350
351
         _closeDelistedMarketPositions(_accounts, _subAccountIds, _marketIndexes,
     tpTokens);
352
         emit LogCloseDelistedMarketPositions(_accounts, _subAccountIds,
353
     _marketIndexes, _tpTokens);
354
```

### Recommendation

 $\neg (A \land B \land C) \equiv \neg A \lor \neg B \lor \neg C$ , use | instead:

```
330
     function closeDelistedMarketPositions(
331
         address[] calldata accounts,
         uint8[] calldata _subAccountIds,
332
         uint256[] calldata _marketIndexes,
333
334
         address[] calldata tpTokens,
         bytes32[] memory priceData,
335
         bytes32[] memory _publishTimeData,
336
337
         uint256 _minPublishTime,
         bytes32 _encodedVaas
338
       ) external payable nonReentrant onlyPositionManager {
339
         // pre-validation
340
         if (
341
           _accounts.length != _subAccountIds.length ||
342
           _subAccountIds.length != _marketIndexes.length ||
343
           _marketIndexes.length != _tpTokens.length
344
         ) revert IBotHandler InvalidArray();
345
346
         // Feed Price
347
         // slither-disable-next-line arbitrary-send-eth
348
         IEcoPyth(pyth).updatePriceFeeds(_priceData, _publishTimeData, _minPublishTime,
349
     _encodedVaas);
350
351
         _closeDelistedMarketPositions(_accounts, _subAccountIds, _marketIndexes,
     _tpTokens);
352
```



```
emit LogCloseDelistedMarketPositions(_accounts, _subAccountIds,
    _marketIndexes, _tpTokens);
}
```





## [WP-G9] EcoPyth.buildPriceUpdateData() Optimizations

Gas

### **Issue Description**

https://github.com/perp88/v2-evm/blob/a6eddbd6a6378da3e9dc6f6f4d10b721fa67ba7e/src/oracles/EcoPyth.sol#L146-L171

```
146
        function buildPriceUpdateData(int24[] calldata _prices) external pure returns
      (bytes32[] memory _updateData) {
         updateData = new bytes32[]( prices.length / MAX PRICE PER WORD + 1);
147
          _updateData[0] = bytes32(uint256(0));
148
149
         for (uint256 i; i < _prices.length; i++) {</pre>
            uint256 outerIndex = i / MAX PRICE PER WORD;
150
            uint256 innerIndex = i % MAX PRICE PER WORD;
151
152
153
            bytes32 partialWord = bytes32(
              abi.encodePacked(
154
155
                innerIndex == 0 ? _prices[i] : int24(0),
156
                innerIndex == 1 ? _prices[i] : int24(0),
                innerIndex == 2 ? _prices[i] : int24(0),
157
                innerIndex == 3 ? _prices[i] : int24(0),
158
                innerIndex == 4 ? _prices[i] : int24(0),
159
                innerIndex == 5 ? _prices[i] : int24(0),
160
                innerIndex == 6 ? _prices[i] : int24(0),
161
162
                innerIndex == 7 ? _prices[i] : int24(0),
163
                innerIndex == 8 ? _prices[i] : int24(0),
                innerIndex == 9 ? _prices[i] : int24(0)
164
165
              )
166
            );
            bytes32 previousWord = _updateData[outerIndex];
167
168
169
            updateData[outerIndex] = previousWord | partialWord;
         }
170
171
       }
```

- L147 should use divUp(uint256 a, uint256 b) { return (a + b 1) / b; } .
  - In the current implementation, when \_prices.length is a multiple of MAX\_PRICE\_PER\_WORD , \_updateData.length will be 1 more than expected.
- L167 L169 can be changed to updateData[outerIndex] |= partialWord; for better



readability and to avoid redundant local variable previousWord.

- L153 L166 the calculation of partialWord can be simplified to bytes32(uint256(uint24(\_prices[i])) < 24 \* (MAX\_PRICE\_PER\_WORD 1 innerIndex) + 16)
- L148 \_updateData[0] = bytes32(uint256(0)); is redundant and peculiar (initializing \_updateData[0] separately while there is no need to initialize it, just like the other \_updateData[i] ).
- L149 i++ can be changed to ++1 for gas optimization.

### Recommendation

Consdier changing to:

```
function buildPriceUpdateData(int24[] calldata _prices) external pure returns
146
      (bytes32[] memory _updateData) {
          _updateData = new bytes32[]((_prices.length + MAX_PRICE_PER_WORD - 1) /
147
     MAX_PRICE_PER_WORD);
          for (uint256 i; i < _prices.length; ++i) {</pre>
148
              uint256 outerIndex = i / MAX PRICE PER WORD;
149
              uint256 innerIndex = i % MAX PRICE PER WORD;
150
              bytes32 partialWord = bytes32(uint256(uint24(_prices[i])) << 24 *</pre>
151
      (MAX_PRICE_PER_WORD - 1 - innerIndex) + 16);
              updateData[outerIndex] |= partialWord;
152
          }
153
154
     }
```

EcoPyth.buildPublishTimeUpdateData() has similar issues.

### **Status**





# [WP-M10] LiquidityHandler.cancelLiquidityOrder()' should refunds the execution fee

#### Medium

## **Issue Description**

LiquidityHandler.cancelLiquidityOrder() does not refund order.executionFee as stated in the function's comment.

This results in the **executionFee** being locked in the contract with no destination (neither refunded to the user nor sent to the executor).

https://github.com/perp88/v2-evm/blob/a6eddbd6a6378da3e9dc6f6f4d10b721fa67ba7e/src/handlers/LiquidityHandler.sol#L391-L418

```
/// @notice Cancels the specified add/remove liquidity order and refunds the
391
     execution fee.
392
       /// @param _orderIndex Index of the order to cancel.
       function cancelLiquidityOrder(uint256 _orderIndex) external nonReentrant {
393
         // if order index >= liquidity order's length, then out of bound
394
395
         // if order index < next execute index, means order index outdate
396
         if (_orderIndex >= liquidityOrders.length || _orderIndex <</pre>
     nextExecutionOrderIndex) {
            revert ILiquidityHandler_NoOrder();
397
         }
398
399
         // SLOAD
400
401
         LiquidityOrder memory _order = liquidityOrders[_orderIndex];
402
         // validate if msg.sender is not owned the order, then revert
403
404
          if (msg.sender != liquidityOrders[ orderIndex].account) revert
     ILiquidityHandler_NotOrderOwner();
405
406
          delete liquidityOrders[ orderIndex];
407
408
          _refund(_order);
409
410
          emit LogCancelLiquidityOrder(
            payable(msg.sender),
411
```



```
412    __order.orderId,
413    __order.token,
414    __order.amount,
415    __order.minOut,
416    __order.isAdd
417    );
418 }
```

https://github.com/perp88/v2-evm/blob/a6eddbd6a6378da3e9dc6f6f4d10b721fa67ba7e/src/handlers/LiquidityHandler.sol#L424-L450

```
424
       /// @notice refund order
       /// @dev this method has not be called directly
425
       /// @param order order to execute
426
       // slither-disable-next-line
427
       function _refund(LiquidityOrder memory _order) private {
428
         // if found order with amount 0. means order has been executed or canceled
429
         uint256 amount = order.amount;
430
         if ( amount == 0) return;
431
432
         address _account = _order.account;
433
434
435
         // Add Liquidity order
         if ( order.isAdd) {
436
437
           if (_order.isNativeOut) {
438
              _transferOutETH(_amount, _account);
439
           } else {
440
              IERC20Upgradeable(_order.token).safeTransfer(_account, _amount);
441
           }
442
            emit LogRefund(_account, _order.orderId, _order.token, _amount,
     _order.isNativeOut);
443
444
         // Remove Liquidity order
         else {
445
           address hlp =
446
     ConfigStorage(LiquidityService(liquidityService).configStorage()).hlp();
           IERC20Upgradeable(hlp).safeTransfer(_account, _amount);
447
           emit LogRefund(_account, _order.orderId, hlp, _amount, false);
448
449
         }
450
       }
```



### Recommendation

**cancelLiquidityOrder()** should refund the **executionFee**. This is because there is no need to actually execute an order that has been cancelled.

https://github.com/perp88/v2-evm/blob/a6eddbd6a6378da3e9dc6f6f4d10b721fa67ba7e/src/handlers/LiquidityHandler.sol#L391-L418

```
391
       /// @notice Cancels the specified add/remove liquidity order and refunds the
     execution fee.
       /// @param _orderIndex Index of the order to cancel.
392
       function cancelLiquidityOrder(uint256 _orderIndex) external nonReentrant {
393
         // if order index >= liquidity order's length, then out of bound
394
         // if order index < next execute index, means order index outdate
395
         if (_orderIndex >= liquidityOrders.length || _orderIndex <</pre>
396
     nextExecutionOrderIndex) {
397
            revert ILiquidityHandler NoOrder();
398
         }
399
         // SLOAD
400
401
         LiquidityOrder memory _order = liquidityOrders[_orderIndex];
402
         // validate if msg.sender is not owned the order, then revert
403
          if (msg.sender != liquidityOrders[_orderIndex].account) revert
404
     ILiquidityHandler_NotOrderOwner();
405
406
          delete liquidityOrders[_orderIndex];
407
         _refund(_order);
408
409
410
         // refund the _order.executionFee
411
          _transferOutETH(_order.executionFee, msg.sender);
412
413
          emit LogCancelLiquidityOrder(
414
            payable(msg.sender),
            order.orderId,
415
            _order.token,
416
417
            _order.amount,
            order.minOut,
418
            order.isAdd
419
420
         );
421
       }
```







# [WP-M11] Cancelled WithdrawOrder should be skipped

#### Medium

## **Issue Description**

https://github.com/perp88/v2-evm/blob/4b1cf356b2ec4cb33955f2b79a852db059070f7b/src/handlers/CrossMarginHandler.sol#L285-L363

```
285
        function executeOrder(
286
          uint256 endIndex,
          address payable _feeReceiver,
287
288
          bytes32[] memory _priceData,
          bytes32[] memory _publishTimeData,
289
          uint256 _minPublishTime,
290
          bytes32 encodedVaas
291
292
        ) external nonReentrant onlyOrderExecutor {
293
          // SLOAD
     @@ 294,319 @@
320
321
          for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {</pre>
322
            order = withdrawOrders[i];
323
            _executionFee = _order.executionFee;
324
            try this.executeWithdrawOrder(_order) {
325
              emit LogExecuteWithdrawOrder(
326
                _order.account,
327
328
                order.subAccountId,
329
                order.orderId,
330
                _order.token,
331
                _order.amount,
                _order.shouldUnwrap,
332
333
                true,
334
335
              );
336
              // update order status
337
              _order.status = WithdrawOrderStatus.SUCCESS;
338
            } catch Error(string memory errMsg) {
339
              _handleOrderFail(_order, errMsg);
            } catch Panic(uint /*errorCode*/) {
340
              _handleOrderFail(_order, "Panic occurred while executing the withdraw
341
     order");
```



```
342
            } catch (bytes memory errMsg) {
343
              _handleOrderFail(_order, string(errMsg));
344
            }
345
            // assign exec time
346
347
            _order.executedTimestamp = uint48(block.timestamp);
            totalFeeReceiver += executionFee;
348
349
350
            // save to executed order first
351
            subAccountExecutedWithdrawOrders[HMXLib.getSubAccount(_order.account,
     _order.subAccountId)].push(_order);
           // clear executed withdraw order
352
            delete withdrawOrders[i];
353
354
            unchecked {
355
356
              ++i;
            }
357
         }
358
359
         nextExecutionOrderIndex = endIndex + 1;
360
         // Pay total collected fees to the executor
361
362
          _transferOutETH(_totalFeeReceiver, _feeReceiver);
363
```

withdrawOrders[ orderIndex]. executionFee was reset to 0 in cancelWithdrawOrder() anyway.

The executor cannot receive any executionFee by attempting to execute the WithdrawOrder.

Furthermore, the **executionFee** must be allocated somewhere, either refunded to the user or given to the executor since they still incurred the gas fee in attempting to execute it.

https://github.com/perp88/v2-evm/blob/4b1cf356b2ec4cb33955f2b79a852db059070f7b/src/handlers/CrossMarginHandler.sol#L421-L446

```
function cancelWithdrawOrder(uint256 _orderIndex) external nonReentrant {
   // if order index >= liquidity order's length, then out of bound
   // if order index < next execute index, means order index outdate
   if (_orderIndex >= withdrawOrders.length || _orderIndex <
        nextExecutionOrderIndex) {
        revert ICrossMarginHandler_NoOrder();
   }
}</pre>
```



```
427
428
          // SLOAD
429
          WithdrawOrder memory _order = withdrawOrders[_orderIndex];
430
431
         // validate if msg.sender is not owned the order, then revert
432
          if (msg.sender != _order.account) revert ICrossMarginHandler_NotOrderOwner();
433
434
          delete withdrawOrders[_orderIndex];
435
          emit LogCancelWithdrawOrder(
436
            payable(msg.sender),
437
            order.subAccountId,
438
            order.orderId,
439
440
            _order.token,
            _order.amount,
441
            _order.executionFee,
442
443
            order.shouldUnwrap
444
         );
445
        }
446
```

#### Recommendation

1. Refund the executionFee:

https://github.com/perp88/v2-evm/blob/4b1cf356b2ec4cb33955f2b79a852db059070f7b/src/handlers/CrossMarginHandler.sol#L421-L446

```
function cancelWithdrawOrder(uint256 _orderIndex) external nonReentrant {
421
         // if order index >= liquidity order's length, then out of bound
422
423
          // if order index < next execute index, means order index outdate
         if (_orderIndex >= withdrawOrders.length || _orderIndex <</pre>
424
     nextExecutionOrderIndex) {
              revert ICrossMarginHandler_NoOrder();
425
426
         }
427
         // SLOAD
428
429
          WithdrawOrder memory _order = withdrawOrders[_orderIndex];
430
         // validate if msg.sender is not owned the order, then revert
431
          if (msg.sender != _order.account) revert ICrossMarginHandler_NotOrderOwner();
432
```



```
433
434
          delete withdrawOrders[_orderIndex];
435
436
         // refund the order.executionFee
          _transferOutETH(_order.executionFee, msg.sender);
437
438
439
          emit LogCancelWithdrawOrder(
              payable(msg.sender),
440
              order.subAccountId,
441
              order.orderId,
442
              _order.token,
443
              _order.amount,
444
              order.executionFee,
445
              _order.shouldUnwrap
446
447
         );
     }
448
449
```

### 1. Require \_amount > 0 in createWithdrawCollateralOrder():

```
240
        function createWithdrawCollateralOrder(
         uint8 subAccountId,
241
242
         address token,
243
         uint256 amount,
244
         uint256 _executionFee,
245
         bool shouldUnwrap
        ) external payable nonReentrant onlyAcceptedToken(_token) returns (uint256
246
     orderId) {
         if (_amount == 0) revert ICrossMarginHandler_BadAmount();
247
         if (_executionFee < minExecutionOrderFee) revert</pre>
248
     ICrossMarginHandler_InsufficientExecutionFee();
         if (msg.value != executionFee) revert
249
     ICrossMarginHandler InCorrectValueTransfer();
250
         if (_shouldUnwrap && _token !=
     ConfigStorage(CrossMarginService(crossMarginService).configStorage()).weth())
            revert ICrossMarginHandler NotWNativeToken();
251
252
     @@ 253,274 @@
         return _orderId;
275
276
       }
```

40



## 1. Skip the cancelled orders:

```
285
        function executeOrder(
          uint256 endIndex,
286
287
          address payable _feeReceiver,
          bytes32[] memory _priceData,
288
          bytes32[] memory _publishTimeData,
289
290
          uint256 _minPublishTime,
          bytes32 encodedVaas
291
292
        ) external nonReentrant onlyOrderExecutor {
293
          // SLOAD
     @@ 294,319 @@
320
321
          for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {</pre>
322
            _order = withdrawOrders[i];
323
            // skip cancelled orders
324
            if (_order.amount == 0) {
325
              unchecked {
326
                  ++i;
327
              }
328
              continue;
329
            }
            _executionFee = _order.executionFee;
330
331
            try this.executeWithdrawOrder(_order) {
332
333
              emit LogExecuteWithdrawOrder(
334
                order.account,
                _order.subAccountId,
335
                _order.orderId,
336
337
                _order.token,
                order.amount,
338
339
                _order.shouldUnwrap,
340
                true,
                0.0
341
342
              );
343
              // update order status
              order.status = WithdrawOrderStatus.SUCCESS;
344
345
            } catch Error(string memory errMsg) {
     @@ 346,350 @@
            }
351
352
```



```
@@ 353,364 @@

365     }
366
367     nextExecutionOrderIndex = _endIndex + 1;
368     // Pay total collected fees to the executor
369     _transferOutETH(_totalFeeReceiver, _feeReceiver);
370 }
```





# [WP-G12] Redundant if (uint8Value > 255) check code

Gas

## **Issue Description**

https://github.com/perp88/v2-evm/blob/4b1cf356b2ec4cb33955f2b79a852db059070f7b/src/libraries/HMXLib.sol#L7-L10

```
function getSubAccount(address _primary, uint8 _subAccountId) internal pure
returns (address _subAccount) {
   if (_subAccountId > 255) revert HMXLib_WrongSubAccountId();
   return address(uint160(_primary) ^ uint160(_subAccountId));
}
```

L8, if (\_subAccountId > 255) revert HMXLib\_WrongSubAccountId(); is unnecessary because uint8 \_subAccountId can never be greater than 255, which is the maximum value for type(uint8).

https://github.com/perp88/v2-evm/blob/4b1cf356b2ec4cb33955f2b79a852db059070f7b/src/handlers/MarketTradeHandler.sol#L276-L279

```
function _getSubAccount(address _primary, uint8 _subAccountId) internal pure
returns (address _subAccount) {
   if (_subAccountId > 255) revert();
   return address(uint160(_primary) ^ uint160(_subAccountId));
}
```

#### Status



43



# [WP-G13] proportionalElapsedInDay() can be simplified.

Gas

## **Issue Description**

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/contracts/Calculator.sol#L972-L979

```
function proportionalElapsedInDay(uint256 _marketIndex) public view returns
      (uint256 elapsed) {
         ConfigStorage _configStorage = ConfigStorage(configStorage);
973
         PerpStorage.Market memory globalMarket =
974
     PerpStorage(perpStorage).getMarketByIndex(_marketIndex);
         uint256 fundingInterval = _configStorage.getTradingConfig().fundingInterval;
975
         uint256 elapsedIntervals = (block.timestamp - globalMarket.lastFundingTime) /
976
     fundingInterval;
         uint256 intervalsInOneDay = 1 days / fundingInterval;
977
         return (elapsedIntervals * 1e18) / intervalsInOneDay;
978
979
```

#### Since

elapsedIntervals = 
$$\frac{\Delta time}{fundingInterval}$$
 and intervalsInOneDay =  $\frac{1days}{fundingInterval}$ , the return value will be

$$\frac{\text{elapsedIntervals} \cdot 1e18}{intervalsInOneDay} = \frac{\frac{\Delta \text{time}}{fundingInterval} \cdot 1e18}{\frac{1 \text{ days}}{fundingInterval}} = \frac{\Delta \text{time} \cdot 1e18}{1 \text{ days}} \, .$$

It indicates that **fundingInterval** could be reduced, so there is no need to include it in the formula.





# [WP-G14] TradeService.increasePosition() redundant HMXLib.abs(\_vars.position.positionSizeE30) > 0 check.

Gas

## **Issue Description**

Because L389 has already ensured that \_vars.position.positionSizeE30 is not 0, therefore HMXLib.abs(\_vars.position.positionSizeE30) must be greater than 0.

https://github.com/perp88/v2-evm/blob/4b1cf356b2ec4cb33955f2b79a852db059070f7b/src/services/TradeService.sol#L221-L489

```
221
       function increasePosition(
222
          address _primaryAccount,
         uint8 _subAccountId,
223
         uint256 _marketIndex,
224
225
         int256 _sizeDelta,
226
         uint256 limitPriceE30
        ) external nonReentrant onlyWhitelistedExecutor {
227
     @@ 228,387 @@
388
         // if the position size is zero after the update, revert the transaction with
389
         if (_vars.position.positionSizeE30 == 0) revert
     ITradeService_BadPositionSize();
         // Ensure that the new absolute position size is greater than zero, but not
390
     smaller than the minimum allowed position size
391
         if (
392
           HMXLib.abs( vars.position.positionSizeE30) > 0 &&
           HMXLib.abs(_vars.position.positionSizeE30) <</pre>
393
     ConfigStorage(configStorage).minimumPositionSize()
394
          ) revert ITradeService_TooTinyPosition();
     @@ 395,488 @@
489
       }
```





# [WP-G15] Unnecessary sload

Gas

## **Issue Description**

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/storages/ConfigStorage.sol#L358-L371

```
358
       function setMarketConfig(
          uint256 _marketIndex,
359
         MarketConfig calldata _newConfig
360
        ) external onlyOwner returns (MarketConfig memory _marketConfig) {
361
          if (_newConfig.increasePositionFeeRateBPS > MAX_FEE_BPS ||
362
     _newConfig.decreasePositionFeeRateBPS > MAX_FEE_BPS)
363
            revert IConfigStorage_MaxFeeBps();
364
          if (_newConfig.assetClass > assetClassConfigs.length - 1) revert
     IConfigStorage_InvalidAssetClass();
          if ( newConfig.initialMarginFractionBPS <</pre>
365
     _newConfig.maintenanceMarginFractionBPS)
            revert IConfigStorage_InvalidValue();
366
367
368
          emit LogSetMarketConfig( marketIndex, marketConfigs[ marketIndex],
     _newConfig);
          marketConfigs[_marketIndex] = _newConfig;
369
370
          return marketConfigs[_marketIndex];
371
```

#### Recommedation

```
358
       function setMarketConfig(
359
         uint256 marketIndex,
360
         MarketConfig calldata _newConfig
361
        ) external onlyOwner returns (MarketConfig memory _marketConfig) {
362
         if (_newConfig.increasePositionFeeRateBPS > MAX_FEE_BPS | |
     _newConfig.decreasePositionFeeRateBPS > MAX_FEE_BPS)
363
            revert IConfigStorage_MaxFeeBps();
         if ( newConfig.assetClass > assetClassConfigs.length - 1) revert
364
     IConfigStorage_InvalidAssetClass();
```



```
if (_newConfig.initialMarginFractionBPS <
    _newConfig.maintenanceMarginFractionBPS)

revert IConfigStorage_InvalidValue();

emit LogSetMarketConfig(_marketIndex, marketConfigs[_marketIndex],
    _newConfig);

marketConfigs[_marketIndex] = _newConfig;

return _newConfig;

}</pre>
```





# [WP-M16] User's balance in tradingStaking may not be decreased as expected due to precision loss.

#### Medium

## **Issue Description**

In the current implementation of <code>TradingStakingHook.sol#onDecreasePosition()</code>, the code compares the stored <code>UserTokenAmount</code> on the <code>tradingStaking</code> contract and skips the withdrawal if the stored amount is less than the <code>amountToWithdraw</code>.

https://github.com/perp88/v2-evm/blob/cda7a5755a4df60bcd5c746986ba8b2802299137/src/services/TradeService.sol#L502-L567

```
function decreasePosition(
502
503
         address _account,
504
         uint8 subAccountId,
         uint256 _marketIndex,
505
         uint256 _positionSizeE30ToDecrease,
506
         address _tpToken,
507
         uint256 limitPriceE30
508
509
       ) external nonReentrant onlyWhitelistedExecutor {
     @@ 510,560 @@
561
         // update position, market, and global market state
562
         _decreasePosition(_marketConfig, _marketIndex, _vars);
563
564
         // Call Trade Service Hook
565
         _decreasePositionHooks(_account, _subAccountId, _marketIndex,
566
     positionSizeE30ToDecrease);
567
```

https://github.com/perp88/v2-evm/blob/cda7a5755a4df60bcd5c746986ba8b2802299137/src/services/TradeService.sol#L1116-L1129

```
function _decreasePositionHooks(
    address _primaryAccount,
    uint256 _subAccountId,
```



```
1119
           uint256 _marketIndex,
1120
           uint256 _sizeDelta
1121
         ) private {
1122
           address[] memory hooks = ConfigStorage(configStorage).getTradeServiceHooks();
1123
           for (uint256 i; i < _hooks.length; ) {</pre>
1124
             ITradeServiceHook(_hooks[i]).onDecreasePosition(_primaryAccount,
       _subAccountId, _marketIndex, _sizeDelta, "");
             unchecked {
1125
1126
               ++i;
1127
             }
1128
           }
1129
         }
```

https://github.com/perp88/v2-evm/blob/cda7a5755a4df60bcd5c746986ba8b2802299137/src/staking/TradingStakingHook.sol#L45-L57

```
function onDecreasePosition(
45
         address _primaryAccount,
46
47
         uint256,
         uint256 _marketIndex,
         uint256 sizeDelta,
49
50
         bytes32
51
       ) external onlyTradeService {
52
         ITradingStaking ts = ITradingStaking(tradingStaking);
53
         uint256 amountToWithdraw = _sizeDelta / 1e12;
54
         if (ts.getUserTokenAmount(_marketIndex, _primaryAccount) >= amountToWithdraw)
55
           ts.withdraw(_primaryAccount, _marketIndex, amountToWithdraw);
56
         }
57
       }
```

However, when increasing the position, there may be precision loss due to the decimal conversion from 1e30 to 1e18 (TradingStakingHook.sol#L41). This can cause ts.getUserTokenAmount(\_marketIndex, \_primaryAccount) to be less than the size of the position, which can result in skipping the withdrawal when closing the position.

As a result, some of the closed positions may continue to receive tradingStaking rewards.

https://github.com/perp88/v2-evm/blob/81022fc8ff0636ff936de837f607f221b4882c56/src/



## staking/TradingStakingHook.sol#L32-L44

```
32
     function onIncreasePosition(
         address primaryAccount,
33
34
         uint256,
         uint256 _marketIndex,
35
36
         uint256 _sizeDelta,
37
         bytes32
       ) external onlyTradeService {
38
         ITradingStaking ts = ITradingStaking(tradingStaking);
39
40
         if (ts.isMarketIndex(_marketIndex)) {
41
           ts.deposit(_primaryAccount, _marketIndex, _sizeDelta / 1e12);
42
         }
43
       }
44
```

### Recommendation

## Change to:

```
45
    function onDecreasePosition(
46
         address _primaryAccount,
47
         uint256,
         uint256 _marketIndex,
         uint256 _sizeDelta,
49
50
         bytes32
       ) external onlyTradeService {
51
52
         ITradingStaking ts = ITradingStaking(tradingStaking);
53
         uint256 amountToWithdraw = _sizeDelta / 1e12;
54
         uint256 userTokenAmount = ts.getUserTokenAmount(_marketIndex,
     _primaryAccount);
         if (userTokenAmount >= amountToWithdraw) {
55
           ts.withdraw( primaryAccount, marketIndex, amountToWithdraw);
56
57
         } else {
           ts.withdraw(_primaryAccount, _marketIndex, userTokenAmount);
58
59
         }
60
       }
```

50







[WP-M17] Malicious users can prevent the execution of executeLiquidity() by transferring 1 wei token to the vaultStorage contract.

#### Medium

## **Issue Description**

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/handlers/LiquidityHandler.sol#L359-L389

```
359
     function executeLiquidity(LiquidityOrder calldata order) external returns
      (uint256 _amountOut) {
360
         // if not in executing state, then revert
         if (msg.sender != address(this)) revert ILiquidityHandler_Unauthorized();
361
362
363
         if ( order.isAdd) {
364
     IERC20Upgradeable(_order.token).safeTransfer(LiquidityService(liquidityService).vaultStorage(
     order.amount);
           _amountOut = LiquidityService(liquidityService).addLiquidity(
365
              order.account,
366
367
             _order.token,
              order.amount,
368
369
              order.minOut
370
           );
371
372
           return _amountOut;
          } else {
373
374
           _amountOut = LiquidityService(liquidityService).removeLiquidity(
              _order.account,
375
             _order.token,
376
              _order.amount,
377
             order.minOut
378
379
           );
380
381
           if (_order.isNativeOut) {
              _transferOutETH(_amountOut, payable(_order.account));
382
           } else {
383
              IERC20Upgradeable(_order.token).safeTransfer(_order.account, _amountOut);
384
```



https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/services/LiquidityService.sol#L127-L182

```
127
     function addLiquidity(
128
          address _lpProvider,
129
          address _token,
130
         uint256 amount,
         uint256 minAmount
131
132
        ) external nonReentrant onlyWhitelistedExecutor onlyAcceptedToken(_token)
      returns (uint256) {
133
         AddLiquidityVars memory vars;
134
         // SLOAD
135
         _vars.configStorage = ConfigStorage(configStorage);
136
         _vars.calculator = Calculator(_vars.configStorage.calculator());
137
138
         // 1. validate
139
          _validatePreAddRemoveLiquidity( amount);
140
141
142
         if (VaultStorage(vaultStorage).pullToken( token) != amount) {
            revert LiquidityService InvalidInputAmount();
143
144
         }
145
146
         // 2. get price for using to join Pool
147
          (_vars.price, ) = OracleMiddleware(_vars.calculator.oracle()).getLatestPrice(
148
            _vars.configStorage.tokenAssetIds(_token),
149
            false
150
          );
151
152
         // 3. get AUM and LpSupply before deduction fee
153
         vars.aumE30 = vars.calculator.getAUME30(true);
154
         _vars.lpSupply = ERC20Upgradeable(_vars.configStorage.hlp()).totalSupply();
155
         // 4. calculate hlp mint amount
156
157
          (_vars.tokenValueUSDAfterFee, _vars.mintAmount) = _joinPool(
```

53



```
158
            _token,
159
            amount,
160
            _vars.price,
161
            lpProvider,
            _minAmount,
162
163
            _vars.aumE30,
            vars.lpSupply
164
165
          );
166
          // 5. mint HLP to lp provider
167
          HLP(_vars.configStorage.hlp()).mint(_lpProvider, _vars.mintAmount);
168
169
170
          if (HLP(_vars.configStorage.hlp()).totalSupply() < 1e18) revert</pre>
      LiquidityService_TinyShare();
171
          emit AddLiquidity(
172
            lpProvider,
173
174
            _token,
175
            _amount,
176
            _vars.aumE30,
177
            _vars.lpSupply,
178
            _vars.tokenValueUSDAfterFee,
179
            vars.mintAmount
180
          );
181
          return _vars.mintAmount;
182
```

https://github.com/perp88/v2-evm/blob/4c7c35768458a51bcf9d314298fba7a79c5c682a/src/storages/VaultStorage.sol#L94-L104

```
function pullToken(address _token) external nonReentrant onlyWhitelistedExecutor
     returns (uint256) {
95
         return _pullToken(_token);
96
       }
97
98
       function _pullToken(address _token) internal returns (uint256) {
99
         uint256 prevBalance = totalAmount[_token];
100
         uint256 nextBalance = IERC20Upgradeable(_token).balanceOf(address(this));
101
         totalAmount[_token] = nextBalance;
102
103
         return nextBalance - prevBalance;
```



```
104 }
```

The execution of addLiquidity() requires that the delta amount returned by VaultStorage(vaultStorage).pullToken(\_token) strictly equals \_amount .

As a result, anyone can transfer 1 wei of \_token to the vaultStorage contract and break it, effectively preventing the add LiquidityOrder from being fulfilled.

#### Recommendation

Change to:

```
function addLiquidity(
127
          address lpProvider,
128
129
         address _token,
         uint256 _amount,
130
131
         uint256 minAmount
132
        ) external nonReentrant onlyWhitelistedExecutor onlyAcceptedToken(_token)
     returns (uint256) {
         AddLiquidityVars memory _vars;
133
134
135
         // SLOAD
         _vars.configStorage = ConfigStorage(configStorage);
136
         _vars.calculator = Calculator(_vars.configStorage.calculator());
137
138
139
         // 1. validate
140
         validatePreAddRemoveLiquidity( amount);
141
         if (VaultStorage(vaultStorage).pullToken(_token) < _amount) {</pre>
142
143
            revert LiquidityService_InvalidInputAmount();
144
         }
145
         // 2. get price for using to join Pool
146
          (_vars.price, ) = OracleMiddleware(_vars.calculator.oracle()).getLatestPrice(
147
148
            _vars.configStorage.tokenAssetIds(_token),
            false
149
150
         );
151
152
         // 3. get AUM and LpSupply before deduction fee
          _vars.aumE30 = _vars.calculator.getAUME30(true);
153
```



```
_vars.lpSupply = ERC20Upgradeable(_vars.configStorage.hlp()).totalSupply();
154
155
156
          // 4. calculate hlp mint amount
          ( vars.tokenValueUSDAfterFee, vars.mintAmount) = joinPool(
157
158
            _token,
159
            _amount,
160
            _vars.price,
            _lpProvider,
161
            minAmount,
162
163
            _vars.aumE30,
164
            _vars.lpSupply
165
          );
166
          // 5. mint HLP to lp provider
167
168
          HLP(_vars.configStorage.hlp()).mint(_lpProvider, _vars.mintAmount);
169
          if (HLP(_vars.configStorage.hlp()).totalSupply() < 1e18) revert</pre>
170
     LiquidityService_TinyShare();
171
          emit AddLiquidity(
172
173
            _lpProvider,
174
            _token,
175
            _amount,
176
            _vars.aumE30,
177
            _vars.lpSupply,
178
            _vars.tokenValueUSDAfterFee,
            _vars.mintAmount
179
180
          );
181
          return _vars.mintAmount;
182
        }
```

**✓** Fixed



# [WP-N19] Dev related codes to be removed

# **Issue Description**

https://github.com/perp88/v2-evm/blob/cda7a5755a4df60bcd5c746986ba8b2802299137/src/handlers/LimitTradeHandler.sol#L22

```
22 import { console2 } from "forge-std/console2.sol";
```

https://github.com/perp88/v2-evm/blob/cda7a5755a4df60bcd5c746986ba8b2802299137/src/storages/VaultStorage.sol#L12

```
import { console2 } from "forge-std/console2.sol";
```





# **Appendix**

## Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.



# Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.