HMX Audit Report

Jun 27, 2023

WATCHPUG

Table of Contents

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 `createAddLiquidityOrder()` to ensure that the caller has paid for the `_executionFee` specified in the calldata when `_shouldWrap` is set to `false` . This allows an attacker to specify an extremely high `executionFee` , potentially all the balance of the `LiquidityHandler` contract.

Although the current implementation of `cancelLiquidityOrder()` does not actually refund the execution fee as stated in the comment, an attacker cannot exploit this by simply calling `createAddLiquidityOrder()` and `cancelLiquidityOrder()` , 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

```
391    /// @notice Cancels the specified add/remove liquidity order and refunds the
execution fee.
392    /// @param _orderIndex Index of the order to cancel.
393    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,
150       uint256 _executionFee,
151       bool _shouldWrap
152     ) external payable nonReentrant onlyAcceptedToken(_tokenIn) returns (uint256
_orderId) {
153       // pre validate
154       LiquidityService(liquidityService).validatePreAddRemoveLiquidity(_amountIn);
```

```
    if (_executionFee < minExecutionOrderFee) revert
ILiquidityHandler_InsufficientExecutionFee();
    if (_shouldWrap && _tokenIn !=
ConfigStorage(LiquidityService(liquidityService).configStorage()).weth())
      revert ILiquidityHandler_NotWNativeToken();

    if (_shouldWrap) {
      if (msg.value != _amountIn + _executionFee) revert
ILiquidityHandler_InCorrectValueTransfer();
    } else {
      if (msg.value != minExecutionOrderFee) revert
ILiquidityHandler_InCorrectValueTransfer();
      IERC20Upgradeable(_tokenIn).safeTransferFrom(msg.sender, address(this),
_amountIn);
    }

    // convert native to WNative (including executionFee)
    _transferInETH();

    _orderId = liquidityOrders.length;

    liquidityOrders.push(
      LiquidityOrder({
        account: payable(msg.sender),
        orderId: _orderId,
        token: _tokenIn,
        amount: _amountIn,
        minOut: _minOut,
        actualAmountOut: 0,
        isAdd: true,
        executionFee: _executionFee,
        isNativeOut: _shouldWrap,
        createdTimestamp: uint48(block.timestamp),
        executedTimestamp: 0,
        status: LiquidityOrderStatus.PENDING
      })
    );

    emit LogCreateAddLiquidityOrder(
@@ 189,195 @@
    );
    return _orderId;
```

```
  }
```

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

```
function executeOrder(
    uint256 _endIndex,
    address payable _feeReceiver,
    bytes32[] calldata _priceData,
    bytes32[] calldata _publishTimeData,
    uint256 _minPublishTime,
    bytes32 _encodedVaas
  ) external nonReentrant onlyOrderExecutor {
@@ 278,302 @@

    for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {
      _order = liquidityOrders[i];
      if (_order.amount > 0) {
        _executionFee = _order.executionFee;

        try this.executeLiquidity(_order) returns (uint256 actualOut) {
@@ 310,329 @@

        // assign exec time
        _order.executedTimestamp = uint48(block.timestamp);
        _totalFeeReceiver += _executionFee;

        // save to executed order first
        accountExecutedLiquidityOrders[_order.account].push(_order);
        // clear executed liquidity order
        delete liquidityOrders[i];
      }

      unchecked {
        ++i;
      }
    }

    nextExecutionOrderIndex = _endIndex + 1;
    // Pay total collected fees to the executor
    _transferOutETH(_totalFeeReceiver, _feeReceiver);
  }
```

Status

✓ **Fixed**

[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

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

```
    }
    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(
    bytes32 _positionId,
    address _subAccount,
    PerpStorage.Position memory _position,
    uint256 _sizeDelta,
    uint32 _positionFeeBPS,
    uint8 _assetClassIndex,
    uint256 _marketIndex
  ) internal returns (uint256 _tradingFee, uint256 _borrowingFee, int256
_fundingFee) {
    // SLOAD
    Calculator _calculator = calculator;

    // Calculate the trading fee
    _tradingFee = (_sizeDelta * _positionFeeBPS) / BPS;
    emit LogSettleTradingFeeValue(_positionId, _subAccount, _tradingFee);

    // Calculate the borrowing fee
    _borrowingFee = _calculator.getBorrowingFee(
      _assetClassIndex,
      _position.reserveValueE30,
      _position.entryBorrowingRate
    );
    // Update global state
    _accumSettledBorrowingFee(_assetClassIndex, _borrowingFee);
    emit LogSettleBorrowingFeeValue(_positionId, _subAccount, _borrowingFee);

    // Calculate the funding fee
    // We are assuming that the market state has been updated with the latest funding rate
    bool _isLong = _position.positionSizeE30 > 0;
    _fundingFee = _calculator.getFundingFee(
      _isLong,
      HMXLib.abs(_position.positionSizeE30),
      PerpStorage(perpStorage).getMarketByIndex(_marketIndex).fundingAccrued,
      _position.lastFundingAccrued
    );

    // Update global state
    _isLong
      ? _updateAccumFundingLong(_marketIndex, -_fundingFee)
      : _updateAccumFundingShort(_marketIndex, -_fundingFee);
    emit LogSettleFundingFeeValue(_positionId, _subAccount, uint256(_fundingFee));

    return (_tradingFee, _borrowingFee, _fundingFee);
  }
```

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);
    PerpStorage.Market memory _market = _perpStorage.getMarketByIndex(_marketIndex);

    _market.accumFundingLong += fundingLong;
    _perpStorage.updateMarket(_marketIndex, _market);
  }

  function _updateAccumFundingShort(uint256 _marketIndex, int256 fundingShort) internal {
    PerpStorage _perpStorage = PerpStorage(perpStorage);
    PerpStorage.Market memory _market = _perpStorage.getMarketByIndex(_marketIndex);

    _market.accumFundingShort += fundingShort;
    _perpStorage.updateMarket(_marketIndex, _market);
  }
```

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 onlyWhitelistedExecutor {
    // SLOAD
    ConfigStorage _configStorage = ConfigStorage(configStorage);
    PerpStorage _perpStorage = PerpStorage(perpStorage);
    VaultStorage _vaultStorage = VaultStorage(vaultStorage);
    OracleMiddleware _oracle = OracleMiddleware(_configStorage.oracle());

    WithdrawFundingFeeSurplusVars memory _vars;

    // 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 paid to trader
    // Loop through all markets to sum funding fee on LONG and SHORT sides
    uint256 len = _configStorage.getMarketConfigsLength();
    for (uint256 i = 0; i < len; ) {
      PerpStorage.Market memory _market = _perpStorage.getMarketByIndex(i);

      if (_market.accumFundingLong < 0) _vars.fundingFeeBookValue += uint256(-_market.accumFundingLong);
      if (_market.accumFundingShort < 0) _vars.fundingFeeBookValue += uint256(-_market.accumFundingShort);

      unchecked {
        ++i;
      }
    }

    // Calculate value of current Funding fee reserve
    _vars.tokenAssetId = _configStorage.tokenAssetIds(_stableToken);
    _vars.tokenDecimal = _configStorage.getAssetTokenDecimal(_stableToken);
    (_vars.tokenPrice, ) = _oracle.getLatestPrice(_vars.tokenAssetId, false);
    _vars.fundingFeeAmount = _vaultStorage.fundingFeeReserve(_stableToken);
    _vars.totalFundingFeeReserveValueE30 = (_vars.fundingFeeAmount *
_vars.tokenPrice) / (10 ** _vars.tokenDecimal);

    // If fundingFeeBookValue > totalFundingFeeReserveValueE30 means protocol has
exceed balance of fee reserved for paying to traders
    // Funding fee surplus = totalFundingFeeReserveValueE30 - fundingFeeBookValue
    if (_vars.fundingFeeBookValue > _vars.totalFundingFeeReserveValueE30 ||
(_vars.totalFundingFeeReserveValueE30 == 0))
      revert ICrossMarginHandler_NoFundingFeeSurplus();

    _vars.fundingFeeSurplusValue = _vars.totalFundingFeeReserveValueE30 -
_vars.fundingFeeBookValue;
    // Transfer surplus amount to HLP
    {
      (uint256 _repayAmount, uint256 _repayValue) = _getRepayAmount(
        _configStorage,
        _oracle,
        _vars.fundingFeeAmount,
        _vars.fundingFeeSurplusValue,
        _stableToken
      );

      _vaultStorage.withdrawSurplusFromFundingFeeReserveToHLP(_stableToken,
_repayAmount);
      _vars.fundingFeeSurplusValue -= _repayValue;
    }

    emit LogWithdrawFundingFeeSurplus(_vars.fundingFeeSurplusValue);
  }
```

Status

✓ Fixed

[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

```
104    function _payPrefund(uint256 missingAccountFunds) internal virtual {
105      if (missingAccountFunds != 0) {
106        (bool success, ) = payable(msg.sender).call{ value: missingAccountFunds,
       gas: type(uint256).max }("");
107        (success);
108        //ignore failure (its EntryPoint's job to verify, not account.)
109      }
110    }
```

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,
466     // allowing the contract to bypass any revert calls from the destination
      address.
467     // By setting the gas limit to 2300, equivalent to the gas limit of the
      transfer method,
468     // the transaction maintains a secure execution."
469        (bool success, ) = _receiver.call{ value: _amountOut, gas: 2300 }("");
470        // shhh compiler
471        success;
472    }
```

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

```
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,
946        // the transaction maintains a secure execution."
947        (bool success, ) = _receiver.call{ value: _amountOut, gas: 2300 }("");
948        // shhh compiler
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

```
359      function executeLiquidity(LiquidityOrder calldata _order) external returns (uint256 _amountOut) {
360        // if not in executing state, then revert
361        if (msg.sender != address(this)) revert ILiquidityHandler_Unauthorized();
362
363        if (_order.isAdd) {
```

```
@@ 364,372 @@
    } else {
      _amountOut = LiquidityService(liquidityService).removeLiquidity(
        _order.account,
        _order.token,
        _order.amount,
        _order.minOut
      );

      if (_order.isNativeOut) {
        _transferOutETH(_amountOut, payable(_order.account));
      } else {
        IERC20Upgradeable(_order.token).safeTransfer(_order.account, _amountOut);
      }

      return _amountOut;
    }
  }
```

Recommendation

Consider sending WNative instead when a native token transfer fails:

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

Status

✓ Fixed

[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

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

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

```
function executeOrder(
    uint256 _endIndex,
    address payable _feeReceiver,
    bytes32[] calldata _priceData,
    bytes32[] calldata _publishTimeData,
    uint256 _minPublishTime,
    bytes32 _encodedVaas
  ) external nonReentrant onlyOrderExecutor {
    uint256 _nextExecutionOrderIndex = nextExecutionOrderIndex;

    // Get the number of liquidity orders
    uint256 _orderLength = liquidityOrders.length;

    // Ensure there are orders to execute
    if (_nextExecutionOrderIndex == _orderLength) revert
ILiquidityHandler_NoOrder();

    // Set the end index to the latest order index if it exceeds the number of
orders
    uint256 _latestOrderIndex = _orderLength - 1;
    if (_endIndex > _latestOrderIndex) {
      _endIndex = _latestOrderIndex;
    }

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
300      LiquidityOrder memory _order;
301      uint256 _totalFeeReceiver;
302      uint256 _executionFee;
303
304      for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {
    @@ 305,340 @@
341        unchecked {
342          ++i;
343        }
344      }
345
346      nextExecutionOrderIndex = _endIndex + 1;
347      // Pay total collected fees to the executor
348      _transferOutETH(_totalFeeReceiver, _feeReceiver);
349    }
```

1. The actual number of executed orders is `_endIndex-_nextExecutionOrderIndex+1` , because `LiquidityHandler.sol` L304 is a closed interval. This also causes `maxExecutionChuck` 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
- `maxExecutionChuck` : 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 `_endIndex` parameter is too large, it is automatically adjusted to execute only `maxExecutionChuck + 1` (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(
271       uint256 _endIndex,
272       address payable _feeReceiver,
273       bytes32[] calldata _priceData,
274       bytes32[] calldata _publishTimeData,
275       uint256 _minPublishTime,
276       bytes32 _encodedVaas
277     ) external nonReentrant onlyOrderExecutor {
278       uint256 _nextExecutionOrderIndex = nextExecutionOrderIndex;
279
280       // Get the number of liquidity orders
281       uint256 _orderLength = liquidityOrders.length;
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;
    if (_endIndex > _latestOrderIndex) {
      _endIndex = _latestOrderIndex;
    }

    // split execution into chunk for preventing exceed block gas limit
    if (_endIndex - _nextExecutionOrderIndex > maxExecutionChunk - 1)
      _endIndex = _nextExecutionOrderIndex + maxExecutionChunk - 1;

    // slither-disable-next-line arbitrary-send-eth
    IEcoPyth(pyth).updatePriceFeeds(_priceData, _publishTimeData, _minPublishTime,
_encodedVaas);

    // Initialize variables for the execution loop
    LiquidityOrder memory _order;
    uint256 _totalFeeReceiver;
    uint256 _executionFee;

    for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {
@@ 305,340 @@
      unchecked {
        ++i;
      }
    }

    nextExecutionOrderIndex = _endIndex + 1;
    // Pay total collected fees to the executor
    _transferOutETH(_totalFeeReceiver, _feeReceiver);
  }
```

Status

✓ Fixed

[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(
    address _primaryAccount,
    uint8 _subAccountId,
    address _tokenOut,
    uint256 _amountIn
) external nonReentrant onlyWhitelistedExecutor returns (uint256 _amountOut) {
    // Get trader's sub-account address
    VaultStorage _vaultStorage = VaultStorage(vaultStorage);
    ConfigStorage _configStorage = ConfigStorage(configStorage);
    _amountOut = ConvertedGlpStrategy(convertedSglpStrategy).execute(_tokenOut,
_amountIn);

    // Adjusting trader balance
    address _subAccount = HMXLib.getSubAccount(_primaryAccount, _subAccountId);
    _vaultStorage.decreaseTraderBalance(_subAccount, _configStorage.sglp(),
_amountIn);
    _vaultStorage.increaseTraderBalance(_subAccount, _tokenOut, _amountOut);

    emit LogConvertSGlpCollateral(_primaryAccount, _subAccountId, _tokenOut,
_amountIn, _amountOut);
    return _amountOut;
  }
```

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

```
  function withdrawCollateral(
    address _primaryAccount,
    uint8 _subAccountId,
    address _token,
    uint256 _amount,
    address _receiver
  ) external nonReentrant onlyWhitelistedExecutor onlyAcceptedToken(_token) {
    // SLOAD
    Calculator _calculator = Calculator(calculator);

    VaultStorage _vaultStorage = VaultStorage(vaultStorage);

    // Get trader's sub-account address
    address _subAccount = HMXLib.getSubAccount(_primaryAccount, _subAccountId);

    // Get current collateral token balance of trader's account
    // and deduct with new token withdrawing amount
    uint256 _oldBalance = _vaultStorage.traderBalances(_subAccount, _token);
    if (_amount > _oldBalance) revert ICrossMarginService_InsufficientBalance();

    // Decrease collateral token balance
    _vaultStorage.decreaseTraderBalance(_subAccount, _token, _amount);

    // Calculate validation for if new Equity is below IMR or not
    int256 equity = _calculator.getEquity(_subAccount, 0, 0);
    if (equity < 0 || uint256(equity) < _calculator.getIMR(_subAccount))
      revert ICrossMarginService_WithdrawBalanceBelowIMR();

    // Transfer withdrawing token from VaultStorage to destination wallet
    _vaultStorage.pushToken(_token, _receiver, _amount);

    emit LogWithdrawCollateral(_primaryAccount, _subAccount, _token, _amount,
_receiver);
  }
```

Status

✓ Fixed

[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.
    bytes memory _callData = abi.encodeWithSelector(
      IGmxRewardRouterV2.unstakeAndRedeemGlp.selector,
      _tokenOut,
      _amount,
      0,
      address(this)
    );

    // 2. withdraw sglp from GMX
    bytes memory _cookResult = vaultStorage.cook(address(sglp), address(rewardRouter), _callData);
    _amountOut = abi.decode(_cookResult, (uint256));

    // 3. Transfer token to vaultStorage
    IERC20Upgradeable(_tokenOut).safeTransfer(address(vaultStorage), _amountOut);
    vaultStorage.pullToken(_tokenOut);

    return _amountOut;
  }
```

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

```
function convertSGlpCollateral(
    address _primaryAccount,
    uint8 _subAccountId,
    address _tokenOut,
    uint256 _amountIn
) external nonReentrant onlyWhitelistedExecutor returns (uint256 _amountOut) {
    // Get trader's sub-account address
    VaultStorage _vaultStorage = VaultStorage(vaultStorage);
    ConfigStorage _configStorage = ConfigStorage(configStorage);
    _amountOut = ConvertedGlpStrategy(convertedSglpStrategy).execute(_tokenOut,
_amountIn);

    // Adjusting trader balance
    address _subAccount = HMXLib.getSubAccount(_primaryAccount, _subAccountId);
    _vaultStorage.decreaseTraderBalance(_subAccount, _configStorage.sglp(),
_amountIn);
    _vaultStorage.increaseTraderBalance(_subAccount, _tokenOut, _amountOut);

    emit LogConvertSGlpCollateral(_primaryAccount, _subAccountId, _tokenOut,
_amountIn, _amountOut);
    return _amountOut;
  }
```

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) {
    require(_glpAmount > 0, "RewardRouter: invalid _glpAmount");

    address account = msg.sender;
    IRewardTracker(stakedGlpTracker).unstakeForAccount(account, feeGlpTracker,
_glpAmount, account);
    IRewardTracker(feeGlpTracker).unstakeForAccount(account, glp, _glpAmount,
account);
    uint256 amountOut = IGlpManager(glpManager).removeLiquidityForAccount(account,
_tokenOut, _glpAmount, _minOut, _receiver);

    emit UnstakeGlp(account, _glpAmount);

    return amountOut;
}
```

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.

Status

✓ **Fixed**

[WP-L8] Wrong implementation of array length validation

Low

Issue Description

1. `BotHandler.sol#deleverages()`

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

```
function deleverages(
    address[] memory _accounts,
    uint8[] memory _subAccountIds,
    uint256[] memory _marketIndexes,
    address[] memory _tpTokens,
    bytes32[] memory _priceData,
    bytes32[] memory _publishTimeData,
    uint256 _minPublishTime,
    bytes32 _encodedVaas
  ) external payable nonReentrant onlyPositionManager {
    // pre-validation
    if (
      _accounts.length != _subAccountIds.length &&
      _subAccountIds.length != _marketIndexes.length &&
      _marketIndexes.length != _tpTokens.length
    ) revert IBotHandler_InvalidArray();

    // Feed Price
    // slither-disable-next-line arbitrary-send-eth
    IEcoPyth(pyth).updatePriceFeeds(_priceData, _publishTimeData, _minPublishTime,
_encodedVaas);

    _deleverages(_accounts, _subAccountIds, _marketIndexes, _tpTokens);

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

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

```
function _deleverages(
    address[] memory _accounts,
    uint8[] memory _subAccountIds,
    uint256[] memory _marketIndexes,
    address[] memory _tpTokens
  ) internal nonReentrant {
    // SLOAD
    TradeService _tradeService = TradeService(tradeService);
    uint256 len = _accounts.length;
    for (uint256 i; i < len; ) {
      _tradeService.validateDeleverage();
      _tradeService.forceClosePosition(_accounts[i], _subAccountIds[i],
_marketIndexes[i], _tpTokens[i]);
      unchecked {
        ++i;
      }
    }
  }
```

2. `BotHandler.sol#closeDelistedMarketPositions()`

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

```
function closeDelistedMarketPositions(
    address[] calldata _accounts,
    uint8[] calldata _subAccountIds,
    uint256[] calldata _marketIndexes,
    address[] calldata _tpTokens,
    bytes32[] memory _priceData,
    bytes32[] memory _publishTimeData,
    uint256 _minPublishTime,
    bytes32 _encodedVaas
  ) external payable nonReentrant onlyPositionManager {
    // pre-validation
    if (
      _accounts.length != _subAccountIds.length &&
      _subAccountIds.length != _marketIndexes.length &&
      _marketIndexes.length != _tpTokens.length
    ) revert IBotHandler_InvalidArray();

    // Feed Price
    // slither-disable-next-line arbitrary-send-eth
    IEcoPyth(pyth).updatePriceFeeds(_priceData, _publishTimeData, _minPublishTime,
_encodedVaas);

    _closeDelistedMarketPositions(_accounts, _subAccountIds, _marketIndexes,
_tpTokens);

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

Recommendation

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

```
function closeDelistedMarketPositions(
    address[] calldata _accounts,
    uint8[] calldata _subAccountIds,
    uint256[] calldata _marketIndexes,
    address[] calldata _tpTokens,
    bytes32[] memory _priceData,
    bytes32[] memory _publishTimeData,
    uint256 _minPublishTime,
    bytes32 _encodedVaas
  ) external payable nonReentrant onlyPositionManager {
    // pre-validation
    if (
      _accounts.length != _subAccountIds.length ||
      _subAccountIds.length != _marketIndexes.length ||
      _marketIndexes.length != _tpTokens.length
    ) revert IBotHandler_InvalidArray();

    // Feed Price
    // slither-disable-next-line arbitrary-send-eth
    IEcoPyth(pyth).updatePriceFeeds(_priceData, _publishTimeData, _minPublishTime,
_encodedVaas);

    _closeDelistedMarketPositions(_accounts, _subAccountIds, _marketIndexes,
_tpTokens);

```

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

Status

✓ **Fixed**

[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) {
147     _updateData = new bytes32[](_prices.length / MAX_PRICE_PER_WORD + 1);
148     _updateData[0] = bytes32(uint256(0));
149     for (uint256 i; i < _prices.length; i++) {
150       uint256 outerIndex = i / MAX_PRICE_PER_WORD;
151       uint256 innerIndex = i % MAX_PRICE_PER_WORD;
152
153       bytes32 partialWord = bytes32(
154         abi.encodePacked(
155           innerIndex == 0 ? _prices[i] : int24(0),
156           innerIndex == 1 ? _prices[i] : int24(0),
157           innerIndex == 2 ? _prices[i] : int24(0),
158           innerIndex == 3 ? _prices[i] : int24(0),
159           innerIndex == 4 ? _prices[i] : int24(0),
160           innerIndex == 5 ? _prices[i] : int24(0),
161           innerIndex == 6 ? _prices[i] : int24(0),
162           innerIndex == 7 ? _prices[i] : int24(0),
163           innerIndex == 8 ? _prices[i] : int24(0),
164           innerIndex == 9 ? _prices[i] : int24(0)
165         )
166       );
167       bytes32 previousWord = _updateData[outerIndex];
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
(bytes32[] memory _updateData) {
    _updateData = new bytes32[]((_prices.length + MAX_PRICE_PER_WORD - 1) /
MAX_PRICE_PER_WORD);
    for (uint256 i; i < _prices.length; ++i) {
        uint256 outerIndex = i / MAX_PRICE_PER_WORD;
        uint256 innerIndex = i % MAX_PRICE_PER_WORD;
        bytes32 partialWord = bytes32(uint256(uint24(_prices[i])) << 24 *
(MAX_PRICE_PER_WORD - 1 - innerIndex) + 16);
        _updateData[outerIndex] |= partialWord;
    }
}
```

`EcoPyth.buildPublishTimeUpdateData()` has similar issues.

Status

✓ Fixed

[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 execution fee.
  /// @param _orderIndex Index of the order to cancel.
  function cancelLiquidityOrder(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 >= liquidityOrders.length || _orderIndex < nextExecutionOrderIndex) {
      revert ILiquidityHandler_NoOrder();
    }

    // SLOAD
    LiquidityOrder memory _order = liquidityOrders[_orderIndex];

    // validate if msg.sender is not owned the order, then revert
    if (msg.sender != liquidityOrders[_orderIndex].account) revert ILiquidityHandler_NotOrderOwner();

    delete liquidityOrders[_orderIndex];

    _refund(_order);

    emit LogCancelLiquidityOrder(
      payable(msg.sender),
      _order.orderId,
      _order.token,
      _order.amount,
      _order.minOut,
      _order.isAdd
    );
  }
```

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

```
  /// @notice refund order
  /// @dev this method has not be called directly
  /// @param _order order to execute
  // slither-disable-next-line
  function _refund(LiquidityOrder memory _order) private {
    // if found order with amount 0. means order has been executed or canceled
    uint256 _amount = _order.amount;
    if (_amount == 0) return;

    address _account = _order.account;

    // Add Liquidity order
    if (_order.isAdd) {
      if (_order.isNativeOut) {
        _transferOutETH(_amount, _account);
      } else {
        IERC20Upgradeable(_order.token).safeTransfer(_account, _amount);
      }
      emit LogRefund(_account, _order.orderId, _order.token, _amount,
_order.isNativeOut);
    }
    // Remove Liquidity order
    else {
      address hlp =
ConfigStorage(LiquidityService(liquidityService).configStorage()).hlp();
      IERC20Upgradeable(hlp).safeTransfer(_account, _amount);
      emit LogRefund(_account, _order.orderId, hlp, _amount, false);
    }
  }
```

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

```
  /// @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 {
    // if order index >= liquidity order's length, then out of bound
    // if order index < next execute index, means order index outdate
    if (_orderIndex >= liquidityOrders.length || _orderIndex < nextExecutionOrderIndex) {
      revert ILiquidityHandler_NoOrder();
    }

    // SLOAD
    LiquidityOrder memory _order = liquidityOrders[_orderIndex];

    // validate if msg.sender is not owned the order, then revert
    if (msg.sender != liquidityOrders[_orderIndex].account) revert ILiquidityHandler_NotOrderOwner();

    delete liquidityOrders[_orderIndex];

    _refund(_order);

    // refund the _order.executionFee
    _transferOutETH(_order.executionFee, msg.sender);

    emit LogCancelLiquidityOrder(
      payable(msg.sender),
      _order.orderId,
      _order.token,
      _order.amount,
      _order.minOut,
      _order.isAdd
    );
  }
```

Status

✓ Fixed

[WP-M11] Cancelled `WithdrawOrder` should be skipped

Medium

Issue Description

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

```
  function executeOrder(
    uint256 _endIndex,
    address payable _feeReceiver,
    bytes32[] memory _priceData,
    bytes32[] memory _publishTimeData,
    uint256 _minPublishTime,
    bytes32 _encodedVaas
  ) external nonReentrant onlyOrderExecutor {
    // SLOAD
@@ 294,319 @@

    for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {
      _order = withdrawOrders[i];
      _executionFee = _order.executionFee;

      try this.executeWithdrawOrder(_order) {
        emit LogExecuteWithdrawOrder(
          _order.account,
          _order.subAccountId,
          _order.orderId,
          _order.token,
          _order.amount,
          _order.shouldUnwrap,
          true,
          ""
        );
        // update order status
        _order.status = WithdrawOrderStatus.SUCCESS;
      } catch Error(string memory errMsg) {
        _handleOrderFail(_order, errMsg);
      } catch Panic(uint /*errorCode*/) {
        _handleOrderFail(_order, "Panic occurred while executing the withdraw
order");
```

```
      } catch (bytes memory errMsg) {
        _handleOrderFail(_order, string(errMsg));
      }

      // assign exec time
      _order.executedTimestamp = uint48(block.timestamp);
      _totalFeeReceiver += _executionFee;

      // save to executed order first
      subAccountExecutedWithdrawOrders[HMXLib.getSubAccount(_order.account,
_order.subAccountId)].push(_order);
      // clear executed withdraw order
      delete withdrawOrders[i];

      unchecked {
        ++i;
      }
    }

    nextExecutionOrderIndex = _endIndex + 1;
    // Pay total collected fees to the executor
    _transferOutETH(_totalFeeReceiver, _feeReceiver);
  }
```

`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();
    }

    // SLOAD
    WithdrawOrder memory _order = withdrawOrders[_orderIndex];

    // validate if msg.sender is not owned the order, then revert
    if (msg.sender != _order.account) revert ICrossMarginHandler_NotOrderOwner();

    delete withdrawOrders[_orderIndex];

    emit LogCancelWithdrawOrder(
      payable(msg.sender),
      _order.subAccountId,
      _order.orderId,
      _order.token,
      _order.amount,
      _order.executionFee,
      _order.shouldUnwrap
    );
  }

```

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 {
    // 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();
    }

    // SLOAD
    WithdrawOrder memory _order = withdrawOrders[_orderIndex];

    // validate if msg.sender is not owned the order, then revert
    if (msg.sender != _order.account) revert ICrossMarginHandler_NotOrderOwner();

    delete withdrawOrders[_orderIndex];

    // refund the _order.executionFee
    _transferOutETH(_order.executionFee, msg.sender);

    emit LogCancelWithdrawOrder(
        payable(msg.sender),
        _order.subAccountId,
        _order.orderId,
        _order.token,
        _order.amount,
        _order.executionFee,
        _order.shouldUnwrap
    );
}

```

1. Require `_amount > 0` in `createWithdrawCollateralOrder()` :

```
  function createWithdrawCollateralOrder(
    uint8 _subAccountId,
    address _token,
    uint256 _amount,
    uint256 _executionFee,
    bool _shouldUnwrap
  ) external payable nonReentrant onlyAcceptedToken(_token) returns (uint256 _orderId) {
    if (_amount == 0) revert ICrossMarginHandler_BadAmount();
    if (_executionFee < minExecutionOrderFee) revert ICrossMarginHandler_InsufficientExecutionFee();
    if (msg.value != _executionFee) revert ICrossMarginHandler_InCorrectValueTransfer();
    if (_shouldUnwrap && _token != ConfigStorage(CrossMarginService(crossMarginService).configStorage()).weth())
      revert ICrossMarginHandler_NotWNativeToken();

@@ 253,274 @@

    return _orderId;
  }
```

1. Skip the cancelled orders:

```
  function executeOrder(
    uint256 _endIndex,
    address payable _feeReceiver,
    bytes32[] memory _priceData,
    bytes32[] memory _publishTimeData,
    uint256 _minPublishTime,
    bytes32 _encodedVaas
  ) external nonReentrant onlyOrderExecutor {
    // SLOAD
@@ 294,319 @@

    for (uint256 i = _nextExecutionOrderIndex; i <= _endIndex; ) {
      _order = withdrawOrders[i];
      // skip cancelled orders
      if (_order.amount == 0) {
        unchecked {
            ++i;
        }
        continue;
      }
      _executionFee = _order.executionFee;

      try this.executeWithdrawOrder(_order) {
        emit LogExecuteWithdrawOrder(
          _order.account,
          _order.subAccountId,
          _order.orderId,
          _order.token,
          _order.amount,
          _order.shouldUnwrap,
          true,
          ""
        );
        // update order status
        _order.status = WithdrawOrderStatus.SUCCESS;
      } catch Error(string memory errMsg) {
@@ 346,350 @@
      }

@@ 353,364 @@
    }

    nextExecutionOrderIndex = _endIndex + 1;
    // Pay total collected fees to the executor
    _transferOutETH(_totalFeeReceiver, _feeReceiver);
  }
```

Status

✓ **Fixed**

[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

✓ Fixed

[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);
    PerpStorage.Market memory globalMarket =
PerpStorage(perpStorage).getMarketByIndex(_marketIndex);
    uint256 fundingInterval = _configStorage.getTradingConfig().fundingInterval;
    uint256 elapsedIntervals = (block.timestamp - globalMarket.lastFundingTime) /
fundingInterval;
    uint256 intervalsInOneDay = 1 days / fundingInterval;
    return (elapsedIntervals * 1e18) / intervalsInOneDay;
  }
```

Since

elapsedIntervals $= \frac{\Delta\text{time}}{fundingInterval}$

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.

Status

✓ Fixed

[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,
223     uint8 _subAccountId,
224     uint256 _marketIndex,
225     int256 _sizeDelta,
226     uint256 _limitPriceE30
227   ) external nonReentrant onlyWhitelistedExecutor {
    @@ 228,387 @@
388     // if the position size is zero after the update, revert the transaction with an error
389     if (_vars.position.positionSizeE30 == 0) revert ITradeService_BadPositionSize();
390     // Ensure that the new absolute position size is greater than zero, but not smaller than the minimum allowed position size
391     if (
392       HMXLib.abs(_vars.position.positionSizeE30) > 0 &&
393       HMXLib.abs(_vars.position.positionSizeE30) < ConfigStorage(configStorage).minimumPositionSize()
394     ) revert ITradeService_TooTinyPosition();
    @@ 395,488 @@
489   }
```

Status

✓ Fixed

[WP-G15] Unnecessary sload

Gas

Issue Description

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

```
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();
364      if (_newConfig.assetClass > assetClassConfigs.length - 1) revert
       IConfigStorage_InvalidAssetClass();
365      if (_newConfig.initialMarginFractionBPS <
       _newConfig.maintenanceMarginFractionBPS)
366        revert IConfigStorage_InvalidValue();
367
368      emit LogSetMarketConfig(_marketIndex, marketConfigs[_marketIndex],
       _newConfig);
369      marketConfigs[_marketIndex] = _newConfig;
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();
364      if (_newConfig.assetClass > assetClassConfigs.length - 1) revert
       IConfigStorage_InvalidAssetClass();
```

```
    if (_newConfig.initialMarginFractionBPS < _newConfig.maintenanceMarginFractionBPS)
      revert IConfigStorage_InvalidValue();

    emit LogSetMarketConfig(_marketIndex, marketConfigs[_marketIndex], _newConfig);
    marketConfigs[_marketIndex] = _newConfig;
    return _newConfig;
  }
```

Status

✓ Fixed

[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 `TradingStakingHook.sol#onDecreasePosition()` , the code compares the stored `UserTokenAmount` on the `tradingStaking` contract and skips the withdrawal if the stored amount is less than the `amountToWithdraw` .

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

```
function decreasePosition(
    address _account,
    uint8 _subAccountId,
    uint256 _marketIndex,
    uint256 _positionSizeE30ToDecrease,
    address _tpToken,
    uint256 _limitPriceE30
  ) external nonReentrant onlyWhitelistedExecutor {
@@ 510,560 @@

    // update position, market, and global market state
    _decreasePosition(_marketConfig, _marketIndex, _vars);

    // Call Trade Service Hook
    _decreasePositionHooks(_account, _subAccountId, _marketIndex,
_positionSizeE30ToDecrease);
  }
```

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; ) {
1124        ITradeServiceHook(_hooks[i]).onDecreasePosition(_primaryAccount, _subAccountId, _marketIndex, _sizeDelta, "");
1125        unchecked {
1126          ++i;
1127        }
1128      }
1129    }
```

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

```
45  function onDecreasePosition(
46      address _primaryAccount,
47      uint256,
48      uint256 _marketIndex,
49      uint256 _sizeDelta,
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

```
function onIncreasePosition(
    address _primaryAccount,
    uint256,
    uint256 _marketIndex,
    uint256 _sizeDelta,
    bytes32
  ) external onlyTradeService {
    ITradingStaking ts = ITradingStaking(tradingStaking);
    if (ts.isMarketIndex(_marketIndex)) {
      ts.deposit(_primaryAccount, _marketIndex, _sizeDelta / 1e12);
    }
  }

```

Recommendation

Change to:

```
function onDecreasePosition(
    address _primaryAccount,
    uint256,
    uint256 _marketIndex,
    uint256 _sizeDelta,
    bytes32
  ) external onlyTradeService {
    ITradingStaking ts = ITradingStaking(tradingStaking);
    uint256 amountToWithdraw = _sizeDelta / 1e12;
    uint256 userTokenAmount = ts.getUserTokenAmount(_marketIndex,
_primaryAccount);
    if (userTokenAmount >= amountToWithdraw) {
      ts.withdraw(_primaryAccount, _marketIndex, amountToWithdraw);
    } else {
      ts.withdraw(_primaryAccount, _marketIndex, userTokenAmount);
    }
  }
```

Status

✓ Fixed

[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
361       if (msg.sender != address(this)) revert ILiquidityHandler_Unauthorized();
362
363       if (_order.isAdd) {
364
      IERC20Upgradeable(_order.token).safeTransfer(LiquidityService(liquidityService).vaultStorage(
      _order.amount);
365         _amountOut = LiquidityService(liquidityService).addLiquidity(
366           _order.account,
367           _order.token,
368           _order.amount,
369           _order.minOut
370         );
371
372         return _amountOut;
373       } else {
374         _amountOut = LiquidityService(liquidityService).removeLiquidity(
375           _order.account,
376           _order.token,
377           _order.amount,
378           _order.minOut
379         );
380
381         if (_order.isNativeOut) {
382           _transferOutETH(_amountOut, payable(_order.account));
383         } else {
384           IERC20Upgradeable(_order.token).safeTransfer(_order.account, _amountOut);
```

```
        }

        return _amountOut;
      }
    }
```

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

```
function addLiquidity(
    address _lpProvider,
    address _token,
    uint256 _amount,
    uint256 _minAmount
  ) external nonReentrant onlyWhitelistedExecutor onlyAcceptedToken(_token)
returns (uint256) {
    AddLiquidityVars memory _vars;

    // SLOAD
    _vars.configStorage = ConfigStorage(configStorage);
    _vars.calculator = Calculator(_vars.configStorage.calculator());

    // 1. validate
    _validatePreAddRemoveLiquidity(_amount);

    if (VaultStorage(vaultStorage).pullToken(_token) != _amount) {
      revert LiquidityService_InvalidInputAmount();
    }

    // 2. get price for using to join Pool
    (_vars.price, ) = OracleMiddleware(_vars.calculator.oracle()).getLatestPrice(
      _vars.configStorage.tokenAssetIds(_token),
      false
    );

    // 3. get AUM and LpSupply before deduction fee
    _vars.aumE30 = _vars.calculator.getAUME30(true);
    _vars.lpSupply = ERC20Upgradeable(_vars.configStorage.hlp()).totalSupply();

    // 4. calculate hlp mint amount
    (_vars.tokenValueUSDAfterFee, _vars.mintAmount) = _joinPool(
      _token,
      _amount,
      _vars.price,
      _lpProvider,
      _minAmount,
      _vars.aumE30,
      _vars.lpSupply
    );

    // 5. mint HLP to lp provider
    HLP(_vars.configStorage.hlp()).mint(_lpProvider, _vars.mintAmount);

    if (HLP(_vars.configStorage.hlp()).totalSupply() < 1e18) revert
LiquidityService_TinyShare();

    emit AddLiquidity(
      _lpProvider,
      _token,
      _amount,
      _vars.aumE30,
      _vars.lpSupply,
      _vars.tokenValueUSDAfterFee,
      _vars.mintAmount
    );
    return _vars.mintAmount;
  }
```

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

```
function pullToken(address _token) external nonReentrant onlyWhitelistedExecutor
returns (uint256) {
      return _pullToken(_token);
    }

    function _pullToken(address _token) internal returns (uint256) {
      uint256 prevBalance = totalAmount[_token];
      uint256 nextBalance = IERC20Upgradeable(_token).balanceOf(address(this));

      totalAmount[_token] = nextBalance;
      return nextBalance - prevBalance;
}
```

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(
    address _lpProvider,
    address _token,
    uint256 _amount,
    uint256 _minAmount
  ) external nonReentrant onlyWhitelistedExecutor onlyAcceptedToken(_token)
returns (uint256) {
    AddLiquidityVars memory _vars;

    // SLOAD
    _vars.configStorage = ConfigStorage(configStorage);
    _vars.calculator = Calculator(_vars.configStorage.calculator());

    // 1. validate
    _validatePreAddRemoveLiquidity(_amount);

    if (VaultStorage(vaultStorage).pullToken(_token) < _amount) {
      revert LiquidityService_InvalidInputAmount();
    }

    // 2. get price for using to join Pool
    (_vars.price, ) = OracleMiddleware(_vars.calculator.oracle()).getLatestPrice(
      _vars.configStorage.tokenAssetIds(_token),
      false
    );

    // 3. get AUM and LpSupply before deduction fee
    _vars.aumE30 = _vars.calculator.getAUME30(true);
    _vars.lpSupply = ERC20Upgradeable(_vars.configStorage.hlp()).totalSupply();

    // 4. calculate hlp mint amount
    (_vars.tokenValueUSDAfterFee, _vars.mintAmount) = _joinPool(
      _token,
      _amount,
      _vars.price,
      _lpProvider,
      _minAmount,
      _vars.aumE30,
      _vars.lpSupply
    );

    // 5. mint HLP to lp provider
    HLP(_vars.configStorage.hlp()).mint(_lpProvider, _vars.mintAmount);

    if (HLP(_vars.configStorage.hlp()).totalSupply() < 1e18) revert
LiquidityService_TinyShare();

    emit AddLiquidity(
      _lpProvider,
      _token,
      _amount,
      _vars.aumE30,
      _vars.lpSupply,
      _vars.tokenValueUSDAfterFee,
      _vars.mintAmount
    );
    return _vars.mintAmount;
  }
```

Status

✓ Fixed

[WP-N19] Dev related codes to be removed

Issue Description

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

```
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";
```

Status

✓ Fixed

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