HMXv2 PR 353 Review

Security Review

Cantina Managed review by:
Riley Holterhus, Lead Security Researcher
Throttle, Security Researcher
0x4non, Associate Security Researcher

November 8, 2023

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

HMXv2 is an innovative pool-based perpetual DEX protocol designed to offer a range of advanced features. It introduces multi-asset collateral support and cross-margin flexibility, providing traders with enhanced options and opportunities.

The protocol incorporates secured measurements, including virtual price impact and funding fees, to ensure the protection of liquidity providers (LPs) from being overly exposed to a single direction. By implementing these measures, HMXv2 aims to create a more resilient and balanced trading environment.

From Oct 16th - Oct 18th the Cantina team conducted a review of HMXv2 on commit hash 1072b9d2. The team identified a total of **6** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 3
- Gas Optimizations: 0
- Informational: 3

3 Findings

3.1 Low Risk

3.1.1 Epoch accounting can be gamed

Severity: Low Risk

Context: PerpStorage.sol#L325-L336

Description: In the new adaptive fee mechanism, each new epoch will result in the `epochLongOI` and `epochShortOI` values being reset to zero. Users could achieve lower fees by intentionally placing their trades immediately after a new epoch starts. This might be undesirable or unfair behavior.

Recommendation: The HMX team has implemented a "sliding window" approach to prevent this. Essentially, the most recent `movingWindowLength` epochs will contribute to the fee calculation, lowering the impact of each individual epoch.

Note that the "sliding window" parameters are configurable values, and should be adjusted carefully. For the `movingWindowInterval` parameter specifically, updating this value could affect which epochs are used in the sliding window, effectively resetting the entire fee accounting.

HMX: As explained, this has been addressed in commit 39900c16.

Cantina Managed: Verified.

3.1.2 Epoch accounting should consider trade directions

Severity: Low Risk

Context: TradeHelper.sol#L1018-L1035, PerpStorage.sol#L299-L323

Description: In the new adaptive fee system, open interest is tracked in epochs using the `epochLongOI` and `epochShortOI` mappings. Essentially, `epochLongOI` adjusts based on actions affecting *overall long* positions, while `epochShortOI` does the same for *overall short* positions. Later on, the adaptive fee is calculated using `_getAdaptiveFeeBps()`, which is invoked with `_position.positionSizeE30 > 0` in the `_isLong` argument:

```
function _getAdaptiveFeeBps(
  bool _isLong,
  int256 _sizeDelta,
  uint256 _marketIndex,
  uint32 _baseFeeBps
) internal view returns (uint32 feeBps) {
  (uint256 askDepth, uint256 bidDepth, uint256 coeffVariants) = orderbookOracle.getData(_marketIndex);
  uint256 epochOI = PerpStorage(perpStorage).getEpochOI(_isLong, _marketIndex);
  bool isBuy = _sizeDelta > 0;
  feeBps = adaptiveFeeCalculator.getAdaptiveFeeBps(
    HMXLib.abs(_sizeDelta) / 1e22,
    epochOI / 1e22,
    isBuy ? askDepth : bidDepth,
    coeffVariants,
    _baseFeeBps,
    maxAdaptiveFeeBps
  );
}
```

By considering the overall position direction, and not the individual trade direction, there may be unintended consequences. For example, decreasing a short position is a directionally long trade, but the `epochShortOI` accumulator will be used for fee calculations. Since `askDepth` and `epochShortOI` are used together, but represent opposite directions of the market, the fee calculations can be very volatile in markets with a large skew in one direction.

Recommendation: Consider adjusting the fee calculation to only consider the direction of each trade:

```
  function _getAdaptiveFeeBps(
-   bool _isLong,
    int256 _sizeDelta,
    uint256 _marketIndex,
    uint32 _baseFeeBps
  ) internal view returns (uint32 feeBps) {
    (uint256 askDepth, uint256 bidDepth, uint256 coeffVariants) = orderbookOracle.getData(_marketIndex);
-   uint256 epochOI = PerpStorage(perpStorage).getEpochOI(_isLong, _marketIndex);
    bool isBuy = _sizeDelta > 0;
+   uint256 epochOI = PerpStorage(perpStorage).getEpochOI(isBuy, _marketIndex);
    feeBps = adaptiveFeeCalculator.getAdaptiveFeeBps(
      HMXLib.abs(_sizeDelta) / 1e22,
      epochOI / 1e22,
      isBuy ? askDepth : bidDepth,
      coeffVariants,
      _baseFeeBps,
      maxAdaptiveFeeBps
    );
  }
```

To further accommodate this change, consider changing the `epochLongOI` and `epochShortOI` mappings to represent "epoch directional volume" (perhaps renamed as `epochVolumeBuy` and `epochVolumeSell`). With this logic, any directionally long trade (increasing a long position, decreasing a short position) will increase `epochVolumeBuy`, while any directionally short trade (increasing a short position, decreasing a long position) will increase `epochVolumeSell`. This will eliminate any path dependence in the accounting, and will more accurately represent the recent trading actions.

HMX: This has been addressed in commit 39900c16.

Cantina Managed: Verified.

3.1.3 `_decreasePosition()` now uses incorrect fee rate

Severity: Low Risk

Context: TradeService.sol#L794

Description: In the `_decreasePosition()` function within the `TradeService`, some of the arguments passed to `tradeHelper.updateFeeStates()` have been changed to support the new adaptive fee logic. One argument has been mistakenly changed from `_marketConfig.decreasePositionFeeRateBPS` to `_marketConfig.increasePositionFeeRateBPS`, meaning the wrong rate will be used when closing positions.

Recommendation: Revert this part of the change, so `_marketConfig.decreasePositionFeeRateBPS` is used in `_decreasePosition()` again.

HMX: Fixed in commit b082fdf4.

Cantina Managed: Verified.

3.2 Informational

3.2.1 Adding custom revert errors in `AdaptiveFeeCalculator` would maintain convention

Severity: Informational

Context: AdaptiveFeeCalculator.sol#L56-L60

Description: Throughout most of the HMX codebase, custom revert errors are used instead of `require()` statements. On the other hand, the new `AdaptiveFeeCalculator` is using two `require()` statements in the `pow()` function.

Recommendation: To more closely follow the codebase convention, consider changing these two `require()` statements to use custom revert errors.

HMX: Fixed in commit b082fdf4.

Cantina Managed: Verified.

3.2.2 Unnecessary casting in `AdaptiveFeeCalculator`

Severity: Informational

Context: AdaptiveFeeCalculator.sol#L41

Description: The following return value is used in the `getAdaptiveFeeBps()` function:

```
return uint32(HMXLib.min(ABDKMath64x64.toUInt(ABDKMath64x64.mul(y, BPS_PRECISION_64x64)), uint256(maxFeeBps)));
```

Since `maxFeeBps` is already defined as a `uint256`, the explicit cast is not necessary.

Recommendation: Remove the unnecessary casting of `maxFeeBps`.

HMX: Fixed in commit b082fdf4.

Cantina Managed: Verified.

3.2.3 `OrderbookOracle` has old comments from `EcoPyth2`

Severity: Informational

Context: OrderbookOracle.sol

Description: The `OrderbookOracle` is structured similarly to the `EcoPyth2` contract, and therefore the two contracts share many code snippets. There are a few comments in the `OrderbookOracle` that reference "prices" and "publish time diffs", which are `EcoPyth2` concepts that are not actually relevant to this new contract.

Recommendation: Update the comments in the `OrderbookOracle` to better match the logic that exists in the contract.

HMX: Fixed in commit b082fdf4.

Cantina Managed: Verified.