SHERLOCK

Security Review For BMX DeFi

Collaborative Audit Prepared For:	**BMX DeFi**
Lead Security Expert(s):	**panprog**
Date Audited:	**July 24 - July 25, 2025**
Final Commit:	**55b0368**

Introduction

BMX provides innovative trading solutions for all assets through various product offerings. One such product is Classic for spot and margin trading, built upon the GMX v1 pool model.

Scope

Repository: morphex-labs/morphex-contracts

Audited Commit: ea9d16b9e5f0f2be04daa8dd01c0bb81c99375a9

Final Commit: 55b03683bd39399a533c32d983e47e827bd5ecb7

Files:

- contracts/core/BasePositionManager.sol
- contracts/core/OrderBook.sol
- contracts/core/PositionManager.sol
- contracts/core/PositionRouter.sol
- contracts/core/Router.sol

Final Commit Hash

55b03683bd39399a533c32d983e47e827bd5ecb7

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	1	0

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue M-1: It is possible to manipulate GLP price by increasing position via `PositionManager`, but decreasing it via direct call to `Vault.decreasePosition` which is possible to do even when leverage trading is disabled.

Source: https://github.com/sherlock-audit/2025-07-bmx-defi-july-24th/issues/6

Summary

When the GLP price is calculated, in the `aum` calculations the shorts upnl is calculated and added to `aum`. The shorts upnl is calculated from the `Vault.globalShortsSizes` and `ShortsTracker.globalShortAveragePrices`. In the normal execution flow (via `PositionManager`), both of these values are be updated on position increases and decreases. See more details in the GMX v1 hack writeup.

The GMX v1 hack involved re-entrancy, which bypassed the `PositionManager` and called the `Vault.increasePosition` directly to increase global short sizes, but keep the global short average prices the same, which inflated the GLP price.

The issue still remains even after re-entrancy is fixed: `Vault.decreasePosition` was actually always callable directly by the user. While `PositionManager` enables and disables the leverage to call it, there is no check whether leverage is enabled in the `Vault.decreasePosition`, the check is only present for the `Vault.increasePosition`.

This allows the same attack but via direct call to `Vault.decreasePosition` to decrease the global shorts size, but keep the global shorts average price the same.

Vulnerability Detail

The attack scenario is as following:

1. Suppose

- `current price = 100`
- `global short size = 1`
- `global short average price = 120`
- `AUM assets = 30`
- `GLP totalSupply = 1`
- `shorts upnl = 1 * (120 - 100) = 20`
- `AUM = assets - shorts upnl = 30 - 20 = 10`.

- `GLP price = 10 / 1 = $10`.

2. Buy 10 GLP for $100 (disregard the fees and slippage).
   - `AUM = 130 - 20 = 110`.
   - `GLP totalSupply = 11`.
3. Increase short position (via `PositionManager`) by 99
   - `global short size = 100`
   - `avg price = 100.2` so that shorts upnl is the same:
   - `shorts upnl = 100 * (100.2 - 100) = 20`
4. Immediately close the short position directly via vault: global short size is back to 1, but avg price is still 100.2
   - `shorts upnl = 1 * (100.2 - 100) = 0.2`. This means that `AUM` has instantly increased by 19.8:
   - `AUM = 130 - 0.2 = 129.8`.
   - `GLP price = 129.8 / 11 = $11.8`
5. Immediately sell 10 GLP for $118.
6. The Attacker's profit is `$118 - $100 = $18` less any fees and slippage.
7. While GLP price is still the same, there are no funds in the vault to pay out the profit to opened short.

The process is more complicated than what the hacker used, but it's still possible to steal funds from the vault. However, the amount stolen is limited to several percentages of the Vault funds and it requires either specific setup (short open interest to be in large profit or loss), or the attacker needs to establish long-term short position himself and wait until it's in the large profit or large loss.

Impact

It's possible to steal several percentages from the Vault funds in certain situations.

Another possible impact is that if any user trades with the vault directly to close positions, this will naturally skew the open interest accounting and GLP price calculations will be slightly off, causing some users to be underpaid and some users overpaid.

Code Snippet

The `Vault` prevents the user from calling the `increasePosition` function directly by requiring the leveraged trading to be enabled. The trading is always disabled, but is turned on in `PositionManager`, then immediately turned off again after calling the `Vault`: https://github.com/sherlock-audit/2025-07-bmx-defi-july-24th/blob/612cd9dce6c75071174b14847ba6a18963a44f65/morphex-contracts/contracts/core/Vault.sol#L563-L564

However, `decreasePosition` doesn't have such check as originally the leveraged trading flag was actually used to put market into close-only state, so closing positions was always allowed:
https://github.com/sherlock-audit/2025-07-bmx-defi-july-24th/blob/612cd9dce6c75071174b14847ba6a18963a44f65/morphex-contracts/contracts/core/Vault.sol#L631-L639

If you search for `isLeverageEnabled` - the only place where it's checked is in `increasePosition`. So the `decreasePosition` can be called directly.

Tool Used

Manual Review

Recommendation

The `Vault.decreasePosition` does have external contract validation via `vaultUtils` contract:
https://github.com/sherlock-audit/2025-07-bmx-defi-july-24th/blob/612cd9dce6c75071174b14847ba6a18963a44f65/morphex-contracts/contracts/core/Vault.sol#L638

The `VaultUtils` contract can be modified to require `Vault.isLeverageEnabled == true` instead of current empty implementation: https://github.com/sherlock-audit/2025-07-bmx-defi-july-24th/blob/612cd9dce6c75071174b14847ba6a18963a44f65/morphex-contracts/contracts/core/VaultUtils.sol#L42-L44

This should prevent this issue as it will no longer be possible to call the `Vault.decreasePosition` directly.

Discussion

daedboi

Thank you for the finding @panprog - here is the implementation:
https://github.com/morphex-labs/morphex-contracts/pull/1/commits/34b568dc0857d96659f7df6dc6d8fab2e9b438ac

lpetroulakis

Fix is good. Other vectors explored:

- All possible re-entrancy cases. There are multiple, but all of them are at the end just before the final `emit`, so at worst the `emit` order will be incorrect. The only one where not is `OrderBook.cancelMultiple` (you can do anything in-between cancels), but it's just a convenience function, so there is no probable way of it being abused.
- Ability to call `Vault.increasePosition` or `Vault.decreasePosition` either directly or in a way to avoid updating `ShortsTracker`. That's where the only issue was found, which has already been fixed. Everywhere else, it is always called together with `ShortsTracker`.

- Verified correctness of main on-chain settings. Everything looks correct.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.