Ostium Security Review

Pashov Audit Group

Conducted by: Said, eeyore, saksham

April 6th 2025 - April 7th 2025

Contents

1. About Pashov Audit Group

Pashov Audit Group consists of multiple teams of some of the best smart contract security researchers in the space. Having a combined reported security vulnerabilities count of over 1000, the group strives to create the absolute very best audit journey possible - although 100% security can never be guaranteed, we do guarantee the best efforts of our experienced researchers for your blockchain protocol. Check our previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **0xOstium/smart-contracts** repository was done by **Pashov Audit Group**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Ostium

Ostium is a decentralized perpetual trading protocol of Real World Assets (RWA). It works across commodities, Forex, cryptocurrencies, and a wide array of long-tail assets.

5. Risk Classification

Severity	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hash - 30de704780857f350aacde97ff16f843236b13ba

fixes review commit hash - b09f3bfd9b88578fae345e962ca5f38c0ef11f89

Scope

The following smart contracts were in scope of the audit:

- `OstiumPairInfos`
- `OstiumPairsStorage`
- `OstiumPriceUpKeep`
- `OstiumPrivatePriceUpKeep`
- `OstiumTrading`
- `OstiumTradingCallbacks`
- `Delegatable`
- `TradingCallbacksLib`
- `interfaces/`

7. Executive Summary

Over the course of the security review, Said, eeyore, saksham engaged with Ostium to review Ostium. In this period of time a total of **3** issues were uncovered.

Protocol Summary

Protocol Name	Ostium
Repository	https://github.com/0xOstium/smart-contracts
Date	April 6th 2025 - April 7th 2025
Protocol Type	Perpetual DEX for RWA

Findings Count

Severity	**Amount**
Medium	1
Low	2
Total Findings	**3**

Summary of Findings

ID	Title	Severity	Status
[M-01]	Wrong collateral refund in liquidation when liqPrice == priceAfterImpact	Medium	Resolved
[L-01]	maxLeverage can be set lower than minLeverage overnight	Low	Resolved
[L-02]	Inconsistent validation of liqMarginThresholdP and maxNegativePnlOnOpenP	Low	Resolved

8. Findings

8.1. Medium Findings

[M-01] Wrong collateral refund in liquidation when `liqPrice == priceAfterImpact`

Severity

Impact: High

Likelihood: Low

Description

When a liquidation is triggered and the Oracle price used results in `liqPrice == priceAfterImpact` during the execution of `executeAutomationCloseOrderCallback()`, the system may incorrectly refund a portion of the user collateral - approximately equal to the `liquidationFee`.

This occurs due to a discrepancy in how `value` and `liqMarginValue` are calculated within the `getTradeValuePure()` function. Under specific conditions (`liqPrice == priceAfterImpact`), `value` can become greater than `liqMarginValue`, even though the position should be fully liquidated.

Within the new `Margin-Based Liquidations` logic, users should not receive any collateral back during liquidation. The entire collateral should be distributed between the `liquidationFee` and the `Vault` to cover losing trade.

However, do to the legacy refund logic that remains in the code:

```
@>      uint256 usdcSentToVault = usdcLeftInStorage - usdcSentToTrader;
        storageT.transferUsdc(address(storageT), address
          (this), usdcSentToVault);
        vault.receiveAssets(usdcSentToVault, trade.trader);
@>      if (usdcSentToTrader > 0) storageT.transferUsdc(address
  (storageT), trade.trader, usdcSentToTrader);
```

With combination to the incorrect calculation of `value` and `liqMarginValue`, the `usdcSentToTrader` returned from the `getTradeValue()` function may end up being roughly equal to the `liquidationFee`, resulting in an unintended refund to the liquidated trader.

Recommendation

Ensure that `usdcSentToTrader` is explicitly set to `0` during liquidation, preventing any collateral refund:

```
if (liquidationFee > 0) {
        storageT.transferUsdc(address(storageT), address(this), liquidationFee);
        vault.distributeReward(liquidationFee);
        emit VaultLiqFeeCharged(orderId, tradeId, trade.trader, liquidationFee);
+
+       usdcSentToTrader = 0;
    }
```

8.2. Low Findings

[L-01] `maxLeverage` can be set lower than `minLeverage` overnight

Inside `_setPairOvernightMaxLeverage`, there is no validation to ensure `overnightMaxLeverage` is not lower than `groups[_pair.groupIndex].minLeverage`. Consider adding this validation to ensure proper configuration of `overnightMaxLeverage`.

[L-02] Inconsistent validation of `liqMarginThresholdP` and `maxNegativePnlOnOpenP`

The `liqMarginThresholdP` value is set via `_setLiqMarginThresholdP()`, which validates against a fixed `MAX_LIQ_MARGIN_THRESHOLD_P`. However, this value directly affects the valid range for `maxNegativePnlOnOpenP`, which is constrained to `value <= 100 - liqMarginThresholdP`.

Because there is no cross-check between these two parameters during updates, changing `liqMarginThresholdP` can indirectly invalidate the current `maxNegativePnlOnOpenP` value, potentially pushing it over its allowed limit.

Recommendation

Introduce a validation in `_setLiqMarginThresholdP()` to ensure that the existing `maxNegativePnlOnOpenP` remains valid after the threshold is updated:

```
-       if (value > MAX_LIQ_MARGIN_THRESHOLD_P) {
+       if
+ (value > MAX_LIQ_MARGIN_THRESHOLD_P || maxNegativePnlOnOpenP > 100 - value) {
            revert WrongParams();
        }
```