Security assessment and code review

Initial Delivery: July 25, 2022

Prepared for:
Shao-Kang Lee | Perpetual Protocol
Yenwen Feng | Perpetual Protocol

Prepared by:
Mikerah Quintyne-Collins | HashCloak Inc
Soumen Jana | HashCloak Inc

Table Of Contents

Executive Summary

Perpetual Protocol building a protocol for enabling decentralized and trustless perpetual futures contracts on Ethereum. At the core of their protocol is a concept called the virtual automated market maker (vAMM) that enables one to mint virtual tokens for accounting purposes with no value. In v2 of their protocol, they leverage the increased capital efficiency guarantees of Uniswap V3 in order to build a more capital efficient vAMM.

The Perpetual Protocol Team engaged HashCloak Inc for an audit of their smart contracts written in Solidity. The audit was done with 2 auditors over a 2 week period, from July 20, 2022 to July 22, 2022. The scope of the audit was:

- diff of perp-lushan repository at commit 5551....2bc5 to previous audit.
- diff of perp-curie-liquidity-mining repository at commit 03cf....a728 to previous audit

These commits added the following features to Perpetual Protocol:

- Add Vault.withdrawAll() to withdraw all free collateral(specified) from vault
- Add Vault.withdrawAllEther() to withdraw all ETH from vault
- Fix rounding issue when liquidating collaterals in full
- Fix collateral value precision and underlying rounding issues
- Added price spread checking when adding liquidity
- Add vePERPRewardDistributor contract for distributing and min-lock PERP as vePERP for the beneficiaries
- Add tests for vePERPRewardDistributor contract

We found a variety of issues ranging from medium to informational.

Severity	Number of findings
Critical	0
High	0
Medium	1
Low	0
Informational	6

Findings

Blacklisted USDC holders may lead to frozen funds

Type: Medium
Files affected: Vault.sol

In USDC contract it's possible to blacklist an account and if someone's account or Vault account is blacklisted then the user will not be able to use perp protocol as USDC is the only token used for collateral.

Impact: In the case of a blacklisted user, their funds on Perpetual Protocol may be frozen and they might be unable to manage their positions. If a vault becomes blacklisted, then the user funds held in that vault contract may be frozen.
Suggestion: Add an edge case for withdrawals, closing positions and liquidating positions in the case that either a user or a vault contract becomes blacklisted

pendingFee is shadowed in _getMakerQuoteAndPendingFee

Type: Informational
Files affected: Orderbook.sol

In `_getMakerQuoteBalanceAndPendingFee`, pendingFee is being shadowed by the new declaration of pendingFee on line 678.

Impact: There is no effect on the variable pendingFee on line 677.
Suggestion: Use pendingFee defined on line 677 to the computation on line 678, instead of redeclaring pendingFee.

Gas Optimization in PerpMath

Type: Informational
Files affected: AccountBalance.sol#L439

On lines 439, while calling `_deregisterBaseToken()` function this function is calling `abs()` in PerpMath.sol#L36. It is using a function to convert data that might lead to extra gas usage.

Impact: It will consume more gas.
Suggestion: Use `return value >= 0 ? value.toUint256() : neg256(value).toUint256();` instead of `return value >= 0 ? uint256(value) : uint256(neg256(value));` it will save gas.

Unnecessary return

Type: Informational
Files affected: AccountBalance.sol#L173

For the `settlePositionInClosedMarket()` function, variable names have already been assigned in the function declaration and have already been assigned within the function. However, these are further returned later on in the function when this is unnecessary.

Impact: Extra code will increase the smart contract size.
Suggestion: Remove this line of code `return (positionNotional, openNotional, realizedPnl, closedPrice);`.

Missing zero check

Type: Informational
Files affected: ClearingHouseConfig.sol#L73 CollateralManager.sol#L130

If the `_maxCollateralTokensPerAccount` is set to 0 then no one will be able to deposit collateral. Also if the `_maxMarketsPerAccount` is set to 0 then no one will be able to create any new markets.

Impact: No one may actually be able to use Perpetual Protocol.
Suggestion: Implement zero check for both functions

Misleading Comments

Type: Informational
Files affected: BaseTokenStorage.sol#L12, ClearingHouseStorage.sol#L8, VaultStorage.sol#L8

Perpetual team follows every comment all over the codebase but for this specific case they mentioned that the `_priceFeedDecimals` in `BaseTokenStorage`, `_quoteToken` and `_uniswapV3Factory` in `ClearingHouseStorage` and `_decimals` and `_settlementToken` in `VaultStorage` should be immutable within the comment but didn't use `immutable` in the declaration of these variables.

Impact: It might unexpectedly affect the contract for storage slots.
Suggestion: Check the mentioned line and if the variables are supposed to be `immutable` make those `immutable` or just modify the comment.

JIT (Just in Time) Liquidity

Type: Informational
JIT attack is a type of MEV attack in which LP provides liquidity with a liquidity range in the same block where they see a profitable swap and after the trade they remove the liquidity. As providing a tight liquidity range most of the token will be used from the attacker he will earn the most of the profit. **A sample attack could be like:** An user wanted to trade 200 ETH for USDC and the attacker sees the transaction and can provide liquidity with a tight liquidity range so that he can get all the fees (0.09%) and after the successful transaction he will remove the liquidity. So for this reason regular LPs will not be getting proper returns.

Impact: The impact that JIT attacks have is that over time, JIT attacks should improve prices for users but may result in losses for LPs on Uniswap v3. As such, since Perpetual Protocol is a user of Uniswap v3, the effects of JIT attacks should be minimal.