Custodia Security

IVX v2 Review

Conducted By: Ali Kalout, Ali Shehab

Contents

1. Disclaimer

A smart contract security review cannot ensure the absolute absence of vulnerabilities. This process is limited by time, resources, and expertise, aiming to identify as many vulnerabilities as possible. We cannot guarantee complete security after the review, nor can we assure that the review will detect every issue in your smart contracts. We strongly recommend follow-up security reviews, bug bounty programs, and on-chain monitoring.

2. Introduction

Custodia conducted a security assessment of IVX’s smart contract.

3. About IVX

Decentralized options AMM tailored for zero days to expiry contracts, with a primary focus on crypto and real-world assets, providing high leverage exposure of up to 200x, all through an industry-leading lucid user experience

4. Risk Classification

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

4.1. Impact

- High: Results in a substantial loss of assets within the protocol or significantly impacts a group of users.
- Medium: Causes a minor loss of funds (such as value leakage) or affects a core functionality of the protocol.
- Low: Leads to any unexpected behavior in some of the protocol's functionalities, but is not critical.

4.2. Likelihood

- High: The attack path is feasible with reasonable assumptions that replicate on-chain conditions, and the cost of the attack is relatively low compared to the potential funds that can be stolen or lost..
- Medium: The attack vector is conditionally incentivized but still relatively likely.
- Low: The attack requires too many or highly unlikely assumptions, or it demands a significant stake by the attacker with little or no incentive.

4.3. Action required for severity levels

- Critical: Must fix as soon as possible
- High: Must fix
- Medium: Should fix
- Low: Could fix

5. Security Assessment Summary

Repository: IVX-FI/ivx-contracts

6. Executive Summary

Throughout the security review, Ali Kalout and Ali Shehab engaged with IVX to review IVX. In this period a total of 28 issues were uncovered.

Findings Count

Severity	**Amount**
Critical	4
High	2
Medium	12
Low	10
Total Finding	**28**

Summary of Findings

ID	Title	Severity	Status
[C-01]	BrokerRepo, ReserveRepo, and RewardsRepo are missing `onlySystem` modifier	Critical	Resolved
[C-02]	Buy Call max reserves are being used for Sell Put positions	Critical	Resolved
[C-03]	`DiemOptions::forceClosePositions` will never work because of an inversed portfolio existence check in `DiemOptions::_validatePortfolio`	Critical	Resolved
[C-04]	Settlement and liquidation will always misbehave and result in wrong results because `_closeAllPositionsContracts` is closing contracts before releasing the position reserves	Critical	Resolved
[H-01]	The swap amount in is being used as the swap amount in `_swapPortfolioCollateralTokens`	High	Acknowledged
[H-02]	The settlement payoff depends on the spot price	High	Resolved
[M-01]	`RewardsTracker::_transferToTreasury` should subtract the unclaimed claimable amount before transferring it to the treasury	Medium	Resolved
[M-02]	Sunset stable tokens are not included in `getTokensPriorityOrdered`	Medium	Resolved
[M-03]	All reserves calculations should be rounded up, in favor of the protocol	Medium	Resolved
[M-04]	IVLP token transfer functions don't return true on successful transfers	Medium	Resolved
[M-05]	Wrong entry fee calculation	Medium	Resolved
[M-06]	Wrong token out used when swapping collateral tokens	Medium	Acknowledged
[M-07]	Liquidation favors the fee splitter over the liquidator, when distributing fees, lowering the liquidation incentive	Medium	Acknowledged
[M-08]	Allow swap failures, so the whole system doesn't get DOSed	Medium	Acknowledged

[M-09]	Wrong calculation of `minPrice` in `calculateBSPrices` and `BlackScholesPrices_Vega_Delta`	Medium	Resolved
[M-10]	Invalid validation in `DiemOptions::_validateCloseOption`, blocking users from closing their positions if the minimum number of contracts was increased	Medium	Resolved
[M-11]	Price feed might return stale prices	Medium	Resolved
[M-12]	`getAmmPnlAndPayOff` might differ from the active positions PnL sum, because of using average `p0`	Medium	Acknowledged
[L-01]	`FeeSplitter::_distributeFees` function doesn't check if there are valid distributions, leading to underflow	Low	Acknowledged
[L-02]	Wrong condition used in `_isLiquidatableMMR`	Low	Acknowledged
[L-03]	Withdraw should revert if `canWithdraw` is false	Low	Resolved
[L-04]	`Feesplitter::splitFees` and `FeeSplitter::collectReceiverFees` should use `getWhitelistedTokens` instead of `getActiveTokens`	Low	Resolved
[L-05]	`RewardsTracker::claimableRewards` returns rewards with wrong decimals	Low	Resolved
[L-06]	Users can force tokens to not be removed	Low	Acknowledged
[L-07]	Unbounded ceiling of amounts	Low	Resolved
[L-08]	Calculating the deviation in `TokenWeightBalancer` while depending on the entry/exit will sometimes result in wrong results	Low	Acknowledged
[L-09]	Missing slippage protection on deposit, withdrawal, and open position	Low	Acknowledged
[L-10]	Static fee is used for swap pools	Low	Acknowledged

7. Findings

7.1. Critical Findings

[C-01] BrokerRepo, ReserveRepo, and RewardsRepo are missing `onlySystem` modifier

Severity:
Critical

Description:
BrokerRepo, ReserveRepo, and RewardsRepo are missing `onlySystem` modifier, allowing any user to update critical state variables

Recommendations:
Add necessary modifiers to the repo setters so they’re not permissionless.

[C-02] Buy Call max reserves are being used for Sell Put positions

---

Severity:
Critical

Description:
When calculating the reserves for SP positions, `_ reserveSellPut` calls `_calculateMaximumTokenReservations`, where `_buy` should be passed as false, but it is passed as true.

```
function _calculateMaximumTokenReservations(
    IWhitelistedTokenRepo _whitelistedTokenRepo,
    IReserveRepo _reserveRepo,
    address _token,
    uint256 _totalSupply,
    bool _buy
) private view returns (uint256 maxBuyReserve) {
    uint256 buyRatio = _whitelistedTokenRepo.getTokenBuyCallRatio(_token);
    uint256 sellRatio = _whitelistedTokenRepo.getTokenSellPutRatio(_token);

    if (_buy) {
        maxBuyReserve = Math.mulDiv(
            _totalSupply,
            buyRatio,
            buyRatio + sellRatio,
            Math.Rounding.Floor
        );
    } else {
        maxBuyReserve = Math.mulDiv(
            _totalSupply,
            sellRatio,
            buyRatio + sellRatio,
            Math.Rounding.Floor
        );
    }
    uint256 _totalUnreservedAmount = _totalSupply -
        _reserveRepo.getTokenTotalReserves(_token);
    maxBuyReserve = Math.min(maxBuyReserve, _totalUnreservedAmount);
}
```

Recommendations:
Pass `_buy` as `false` when calculating the reserves of an SP position.

[C-03] `DiemOptions::forceClosePositions` will never work because of an inversed portfolio existence check in `DiemOptions::_validatePortfolio`

Severity:
Critical

Description:
When force-closing positions, the protocol checks if the target portfolio exists; if not, it reverts. However, the condition is inversed. If the portfolio exists, the TX reverts, forcing it to never work.

```
function _validatePortfolio(
    address _portfolio
) private view returns (address) {
    IPortfolioOrganizer _portfolioOrganizer = IPortfolioOrganizer(
        _getContractAddress(PORTFOLIO_ORGANIZER_CONTRACT)
    );

    if (_portfolioOrganizer.checkPortfolioExistence(_portfolio)) {
        revert SenderHasNoPortfolio(msg.sender);
    }

    _checkPortfolioLiquidation(_portfolio);
    return _portfolio;
}
```

Recommendations:
Inverse the existence check:

```
function _validatePortfolio(
    address _portfolio
) private view returns (address) {
    IPortfolioOrganizer _portfolioOrganizer = IPortfolioOrganizer(
        _getContractAddress(PORTFOLIO_ORGANIZER_CONTRACT)
    );

-   if (_portfolioOrganizer.checkPortfolioExistence(_portfolio)) {
+   if (!_portfolioOrganizer.checkPortfolioExistence(_portfolio)) {
        revert SenderHasNoPortfolio(msg.sender);
    }

    _checkPortfolioLiquidation(_portfolio);
    return _portfolio;
}
```

[C-04] Settlement and liquidation will always misbehave and result in wrong results because `_closeAllPositionsContracts` is closing contracts before releasing the position reserves

---

Severity:
Critical

Description:
Both settlement and liquidation call `_closeAllPositionsContracts`, to close all the position's contracts and to release that position reserves. However, there's a critical issue: it closes all the contracts before releasing the reserves, and the releasing uses `_contracts.optionsRepo.getPositionTotalContracts(_positionId)`. `totalNumberOfContracts` will always have a misleading and wrong value as it'll be wrongly decreased, and in case the position was the last one, `totalNumberOfContracts` will be 0 and the TX will revert with "division by 0 error".

Proof of Concept:
```
function test_SettleDOS() public {
        vm.startPrank(bob);
        address portfolio = portfolioOrganizer.createPortfolio();
        WBTC.transfer(portfolio, 1e8);
        WETH.transfer(portfolio, 1e18);
        diemOptions.openPosition(
                OpenPositionParams({
                        token: address(WBTC),
                        strikePrice: 65_000e18,
                        expireDate: block.timestamp + ONE_DAY,
                        positionAction: PositionAction.BUY,
                        positionType: PositionType.CALL,
                        numberOfContracts: 1_00
                })
        );
        vm.stopPrank();
        vm.warp(block.timestamp + epochRepo.getCurrentEpochEndTimestamp());
        _setAvgPrices();
        vm.prank(coordinator);
        optionsManager.setEpoch(block.timestamp + ONE_DAY);
        vm.prank(alice);
        vm.expectRevert(stdError.divisionError); // panic: division or modulo by zero (0x12)
        settler.settlerPortfolio(portfolio);
}
```

Recommendations:

This can be fixed by either of the following:

- Move the reserve releasing logic in `_closeAllPositionsContracts` to be before closing contracts (`_closePositionContracts`).
- Cache _contracts.optionsRepo.getPositionTotalContracts(`_positionId`) before closing contracts, and use it when releasing the position reserves.

7.2. High Findings

[H-01] Swap amount in is being used as the swap amount in `_swapPortfolioCollateralTokens`

Severity:
High

Description:
When a swap happens in `_swapPortfolioCollateralTokens`, the protocol expects an `amountToSwap` to be sent to the receiver, and `amountToSwap` is subtracted according to the amount in for the swap. When swapping an X amount of token into Y amount of token out, the USD value of X is >= USD value of Y, however, the protocol assumes that they're always equal.

A loss will occur but won't be registered/saved, this is because you're using the amount In USD as the amount that will be sent to the receiver (let's say it's the vault in the `collectLossFromPortfolio` case), the protocol will register that it received X USD (amount it), while it'll receive Y USD (amount out), so X - Y is loss that the vault suffered but without it being registered.

Recommendations:
`_amountToSwap` should be subtracted according to the USD value of the swapped amount out resulting from the "exact in" swap.

[H-02] The settlement payoff depends on the spot price

Severity:
High

Description:
The settlement payoff depends on the spot price, where it should be constant as the factors should only depend on the epoch that has passed.

Proof of Concept:
Case 1:
Avg entry price: 460 $
Avg price: 65k $
Spot price: 50k $
Strike price: 65k $
Payoff: -350 $

Case 2:
Avg entry price: 460 $
Avg price: 65k $
Spot price: 70k $
Strike price: 65k $
Payoff: -495 $

7.3. MediumFindings

[M-01] `RewardsTracker::_transferToTreasury` should subtract the unclaimed claimable amount before transferring it to the treasury

---

Severity:
Medium

Description:
User can recalibrate their rewards and claim them later. These unclaimed claimable rewards aren't taken into consideration when transferring the rewards to the treasury `RewardsTracker::_transferToTreasury`.

Recommendations:
When transferring the rewards to the treasury, the unclaimed claimable rewards should be subtracted from the total balance.

[M-02] Sunset stable tokens are not included in `TokenWeightBalancer::getTokensPriorityOrdered`

Severity:
Medium

Description:
When `getTokensPriorityOrdered` is called for exit L236, `_sunsetStableTokens` is computed but never used and not concatenated to the returned `_sunsetTokens`.

Recommendations:
Concat and return the computed sunset stable tokens in `TokenWeightBalancer::getTokensPriorityOrdered`.

[M-03] All reserves calculations should be rounded up, in favor of the protocol

---

Severity:
Medium

Description:
All reserves calculations should be rounded up, in favor of the protocol, in `_reserveBuyPut`, `_reserveSellPut`, and `_reserveSellCall`.

- https://github.com/IVX-FI/ivx-contracts/blob/main/src/ivlp/core/PoolReserver.sol#L322-L339
- https://github.com/IVX-FI/ivx-contracts/blob/main/src/ivlp/core/PoolReserver.sol#L394-L402
- https://github.com/IVX-FI/ivx-contracts/blob/main/src/ivlp/core/PoolReserver.sol#L434-L442

[M-04] IVLP token transfer functions don't return true on successful transfers

---

Severity:
Medium

Description:
IVLP token transfer functions don't return true on successful transfer.

Recommendations:
```
function transfer(address _to, uint256 _value) public override returns (bool) {
    _recalibrateRewards(_msgSender());
    _recalibrateRewards(_to);
-   super.transfer(_to, _value);
+   return super.transfer(_to, _value);
}
function transferFrom(address _from, address _to, uint256 _value) public override returns
(bool) {
    _recalibrateRewards(_from);
    _recalibrateRewards(_to);

-   super.transferFrom(_from, _to, _value);
+   return super.transferFrom(_from, _to, _value);
}
```

[M-05] Wrong entry fee calculation

Severity:
Medium

Description:
The user should pay a fixed fee if the amount he adds is less than or equal to the target weight. However, if he is adding an amount that makes the real weight > target weight more fees should be taken from the user. This is not happening in the code (`_calculateEntryFees`) due to the calculation of the new fee factor.

Recommendations:
Have a minimum amount of entry fee that should be paid by the users.

[M-06] Wrong token out used when swapping collateral tokens

Severity:
Medium

Description:
According to Bob and the docs, the remaining collateral tokens will be swapped to the most unbalanced token (the docs mention that it has to be stable, but Bob confirmed that it could also be tradable).
https://github.com/IVX-FI/ivx-contracts/blob/main/src/portfolio-management/libs/AccountantResolverLib.sol#L137
`_collectableTokens` is indeed sorted, but after cutting losses, the balances could change, and the swap step could be swapping for a token that is balanced or not the most unbalanced token.
https://github.com/IVX-FI/ivx-contracts/blob/main/src/portfolio-management/libs/AccountantResolverLib.sol#L228

It's just using the first stable token, regardless of the balance of that token.

[M-07] Liquidation favors the fee splitter over the liquidator, when distributing fees, lowering the liquidation incentive

---

Severity:
Medium

Description:
Upon liquidation, fee distribution favors the fee splitter cut over the liquidator, which can lower the incentive for liquidation. When liquidating a portfolio whose weight is just over the loss, the liquidator will get nothing in return, as it'll try to fulfill the fee splitter cut first.

https://github.com/IVX-FI/ivx-contracts/blob/main/src/portfolio-management/libs/PortfolioResolverLib.sol#L152-L180

Recommendations:
Favor the liquidator over the fee splitter, to increase the liquidation incentive.

[M-08] Allow swap failures, so the whole system doesn't get DOSed

---

Severity:
Medium

Description:
Swapper executes different swaps, it also provides a strict deadline and a slippage factor, which is correct, however, this makes the swap vulnerable to reverting, which is okay and the swap should revert if 1 of the 2 conditions were violated (slippage and deadline). But the issue here is that if 1 swap reverts, all the TX reverts, i.e. making `collectLossFromPortfolio` and `collectFeesFromPortfolio`, which are used on multiple occurrences, vulnerable to DOS.

The protocol should allow swaps to fail, by wrapping swaps in a try/catch block, and if a swap fails the funds should be sent back to the portfolio. That way we make sure that certain actions, like settlement and liquidation, can't be DOSed, and in the worst-case scenario some debt will be accumulated.

[M-09] Wrong calculation of `minPrice` in `calculateBSPrices` and `BlackScholesPrices_Vega_Delta`

Severity:
Medium

Description:
`minPrice` is calculated as part of the spot price by dividing the spot price by `minPriceFactor` without dividing the answer by the basis scale. The answer is also divided by a very high decimal in `multiplyDecimalRoundPrecise`, forcing it to always be 0 or 1.

- https://github.com/IVX-FI/ivx-contracts/blob/main/src/options/utils/libs/pricer/Pricer.sol#L290-L292
- https://github.com/IVX-FI/ivx-contracts/blob/main/src/options/utils/libs/pricer/Pricer.sol#L356-L358

Recommendations:
```
uint256 minPrice = ((spotPrecise * optionParams.minPriceFactor) / BASIS_POINTS_DIVISOR).preciseDecimalToDecimal();
```

[M-10] Invalid validation in `DiemOptions::_validateCloseOption`, blocking users from closing their positions if the minimum number of contracts was increased

---

Severity:
Medium

Description:
`DiemOptions::_validateCloseOption` checks if the user is closing contracts less than the minimum number of contracts (https://github.com/IVX-FI/ivx-contracts/blob/main/src/options/core/DiemOptions.sol#L763-L768). However, this validation is invalid because it blocks users who have positions

opened with the minimum number of contracts from closing their positions if the coordinator increases the minimum number of contracts.

Recommendations:
Remove that validation, and allow the user to close as many contracts as he wants. The validation should only be done on the remaining contracts in the position.

[M-11] Price feed might return stale prices

Severity:
Medium

Description:
`PriceFeed::getTokenPrice` might return stale prices for tokens, it should:

- Check the `updatedAt` of the latest round data, each feed has a different stale period.
- Check if the price is > 0.

[M-12] `getAmmPnlAndPayOff` might differ from the active positions PnL sum, because of using average `p0`

Severity:
Medium

Description:
`getAmmPnlAndPayOff` might differ from the active positions PnL sum, because of using average `p0`, which affects the minted shares when depositing into the Broker.

7.4. Low Findings

[L-01] `FeeSplitter::_distributeFees` function doesn't check if there are valid distributions, leading to underflow

---

Severity:
Low

Description:
`FeeSplitter::_distributeFees`, loops over the length of the distribution - 1, however, it doesn't check if there are distributions. In case of no distributions, any TX that targets the `_distributeFees` function will revert with an underflow error.
A zero-length check should be added before looping over the distributions.

[L-02] Wrong condition used in `_isLiquidatableMMR`

Severity:
Low

Description:
According to the docs, a portfolio is liquidatable if MMR rises above some threshold. However, according to `_isLiquidatableMMR`, a portfolio is liquidatable if its MMR is greater or equal to the threshold, contradicting the docs.

[L-03] Withdraw should revert if `canWithdraw` is false

Severity:
Low

Description:
`PortfolioOrganizer::withdraw` should revert if `canWithdraw` is false, following the "fail-early and fail-loud" convention.

[L-04] `Feesplitter::splitFees` and `FeeSplitter::collectReceiverFees` should use `getWhitelistedTokens` instead of `getActiveTokens`

Severity:
Low

Description:
`Feesplitter::splitFees` and `FeeSplitter::collectReceiverFees` should use `getWhitelistedTokens` instead of `getActiveTokens`, as active tokens don't contain sunsetted ones, while sunsetted ones can still accumulate fees, on withdrawal for example.

[L-05] `RewardsTracker::claimableRewards` returns rewards with wrong decimals

Severity:
Low

Description:
`RewardsTracker::claimableRewards` returns rewards with wrong decimals, where the answer is not rounded to precise reward token's decimals.

[L-06] Users can force tokens to not be removed

Severity:
Low

Description:
In the pool, after the token has finished the sunset state and the admin decides to remove the token, it will check if the balance is 0 to be able to remove it.

https://github.com/IVX-FI/ivx-contracts/blob/main/src/ivlp/core/Pool.sol#L302-L305

Since we are getting the value by checking the balance, an attacker can send 1 wei to the `REWARDS_VAULT_CONTRACT` causing it always to revert whenever the admin tries to remove the token.

[L-07] Unbounded ceiling of amounts

Severity:
Low

Description:
All ceiled computation of amounts should be bounded to the portfolio's balance, so the TX doesn't revert unexpectedly.

- https://github.com/IVX-FI/ivx-contracts/blob/main/src/portfolio-management/libs/AccountantResolverLib.sol#L93-L98
- https://github.com/IVX-FI/ivx-contracts/blob/main/src/portfolio-management/libs/AccountantResolverLib.sol#L306-L311
- https://github.com/IVX-FI/ivx-contracts/blob/main/src/portfolio-management/libs/AccountantResolverLib.sol#L395-L400

[L-08] Calculating the deviation in `TokenWeightBalancer` while depending on the entry/exit will sometimes result in wrong results

---

Severity:
Low

Description:
Calculating the deviation in `TokenWeightBalancer` while depending on the entry/exit will sometimes result in wrong results, this is because it is not always looking into the best options.

[L-09] Missing slippage protection on deposit, withdrawal, and open position

---

Severity:
Low

Description:
Slippage protection should be added to give users higher control over their actions.

[L-10] Static fee is used for swap pools

Severity:
Low

Description:
In Swapper, use the pre-set pool fee for the pair, and have `3000` as the fallback/default value, https://github.com/IVX-FI/ivx-contracts/blob/main/src/portfolio-management/core/Swapper.sol#L85.