Cove Security Review

Pashov Audit Group

Conducted by: unforgiven, ZeroTrust01, 0x37, Wellbyt3, AresAudits, Opeyemi

April 16th 2025 - April 19th 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 **Storm-Labs-Inc/cove-contracts-core** repository was done by **Pashov Audit Group**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Cove

Cove is a system for automated asset management that maximizes returns by using Programmatic Orders to rebalance portfolios efficiently, eliminating loss-versus-rebalancing (LVR) through off-chain trade matching and optimized execution. It leverages ERC-4626 tokenized vaults to aggregate deposits, minimize slippage and MEV, and provide expert-curated strategies for seamless onchain yield generation.

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 - b791eb90714b724c005ffb10db28df8b2b837c4f

fixes review commit hash - c0011630afa02078cc6c2bbe48b5c46e69769f64

Scope

The following smart contracts were in scope of the audit:

- `AssetRegistry`
- `BasketManager`
- `BasketToken`
- `FeeCollector`
- `BasketManagerUtils`
- `AnchoredOracle`
- `ChainedERC4626Oracle`
- `CurveEMAOracleUnderlying`
- `ERC4626Oracle`
- `ManagedWeightStrategy`
- `StrategyRegistry`
- `CoWSwapAdapter`
- `CoWSwapClone`
- `BasketManagerStorage`

7. Executive Summary

Over the course of the security review, unforgiven, ZeroTrust01, 0x37, Wellbyt3, AresAudits, Opeyemi engaged with Storm Labs to review Cove. In this period of time a total of **3** issues were uncovered.

Protocol Summary

Protocol Name	Cove
Repository	https://github.com/Storm-Labs-Inc/cove-contracts-core
Date	April 16th 2025 - April 19th 2025
Protocol Type	Yield farming management

Findings Count

Severity	**Amount**
Low	3
Total Findings	**3**

Summary of Findings

ID	Title	Severity	Status
[L-01]	ERC4626Oracle and ChainedERC4626Oracle susceptible to price manipulation	Low	Acknowledged
[L-02]	Management fees not updated timely in setManagementFee()	Low	Resolved
[L-03]	Missing fee update in completeRebalance()	Low	Acknowledged

8. Findings

8.1. Low Findings

[L-01] `ERC4626Oracle` and `ChainedERC4626Oracle` susceptible to price manipulation

Code uses `convertToAssets()` and `convertToShares()` to calculate token conversion amounts. The issue is that ERC4626 vaults share price can be manipulated by donation attacks and as result, attackers can perform price manipulation attacks. Performing the attack is costly and the attack would be profitable only in special circumstances like using ERC4626 as collateral.

[L-02] Management fees not updated timely in `setManagementFee()`

In Cove, the management fees will be calculated via `currentTotalSupply`, `feeBps`, and `timeSinceLastHarvest`. In the function `setManagementFee`, we will change the management fee, but we don't accrue the previous time slot's management fees. This will cause incorrect management fee calculation.

```
function setManagementFee(
      addressbasket,
      uint16managementFee_
    ) external onlyRole(_TIMELOCK_ROLE
        if (managementFee_ > _MAX_MANAGEMENT_FEE) {
            revert InvalidManagementFee();
        }

        if (basket != address(0)) {
            uint256 indexPlusOne = _bmStorage.basketTokenToIndexPlusOne[basket];
            if (indexPlusOne == 0) {
                revert BasketTokenNotFound();
            }
            if ((_bmStorage.rebalanceStatus.basketMask &
              (1 << indexPlusOne - 1)) != 0) {
                revert MustWaitForRebalanceToComplete();
            }
        }
        emit ManagementFeeSet
          (basket, _bmStorage.managementFees[basket], managementFee_);
        _bmStorage.managementFees[basket] = managementFee_;
    }
```

```
function _harvestManagementFee
      (uint16 feeBps, address feeCollector) internal {
        uint256 fee = FixedPointMathLib.fullMulDiv(
            currentTotalSupply,
            feeBps * timeSinceLastHarvest,
            ((_MANAGEMENT_FEE_DECIMALS - feeBps) * uint256(365 days))
        );
    }
```

Recommendation: Harvest the management fee before we update the management fees.

[L-03] Missing fee update in `completeRebalance()`

In Cove, we will rebalance the basket's asset weight periodically. In the rebalance process, we may trigger below 4 steps:

1. propose rebalance in timestamp X.
2. propose swap, and finish the internal trade in timestamp X + 1000.
3. execute the swap in timestamp X + 2000.
4. complete the rebalance in timestamp X + 3000. Notice: Anyone can trigger this function.

We can find out that in step1, we will trigger `_harvestManagementFee` to accrue the management fees before `timestamp X`. However, in the function

`completeRebalance`, we don't accrue the management fee between timestamp X and timestamp X + 3000.

In function `completeRebalance`, we have one check `block.timestamp - self.rebalanceStatus.timestamp < self.stepDelay`. The default `stepDelay` is `15 minutes`. The maximum value is `1 hours`. It means that we may miss accruing around 15 minutes of management fees before we finish the rebalancing process.

Impact:

1. When we try to finish the rebalance, we will calculate the pending redeem share's assets amount according to the share's price. Because we don't acquire the last time slot's management fees, the `totalSupply` will be less than the expected value. (When we accrue management fees, we will mint some shares for the fee recipient). The redeemers can avoid paying the last time slot fees.
2. When we try to accrue the management fee next time, we will re-calculate the management fees between timestamp X and timestamp X + 3000. And we will use the latest `totalSupply`. If we have one large pending redeem in above stamp 4, the current `totalSupply` will be less than the actual total supply we should use to calculate the management fees between timestamp X and timestamp X + 3000. The protocol will miss one part of the expected management fees.

```
function proposeRebalance
      (BasketManagerStorage storage self, address[] calldata baskets) external {
        (uint256 pendingDeposits, uint256 pendingRedeems) =
            BasketToken(basket).prepareForRebalance
              (self.managementFees[basket], feeCollector);
    }
    function prepareForRebalance(
        uint16 feeBps,
        address feeCollector
    ) {
        _harvestManagementFee(feeBps, feeCollector);
    }
    function completeRebalance(
        BasketManagerStorage storage self,
        ExternalTrade[] calldata externalTrades,
        address[] calldata baskets,
        uint64[][] calldata basketTargetWeights,
        address[][] calldata basketAssets
    )
        if (block.timestamp - self.rebalanceStatus.timestamp < self.stepDelay) {
            revert TooEarlyToCompleteRebalance();
        }
    }
```

```
function _finalizeRebalance(
        BasketManagerStorage storage self,
        EulerRouter eulerRouter,
        address[] calldata baskets,
        address[][] calldata basketAssets
    ) {
        if (pendingRedeems > 0) {
            uint256 withdrawAmount = eulerRouter.getQuote(
                FixedPointMathLib.fullMulDiv(
                  basketValue,
                  pendingRedeems,
                  BasketToken
                ).totalSupply(
                // _USD_ISO_4217_CODE --> 840
                _USD_ISO_4217_CODE,
                baseAsset
            );
    }
    }
```

Accure the management fees in function `completeRebalance`.