yAudit Olympus Emission Manager Review

Review Resources:

- Bond Protocol Documentation
- The repository

Auditors:

- Jackson
- Invader-tak

Table of Contents

Review Summary

Olympus Emission Manager

The Olympus Emission Manager is intended to issue new Ohm in exchange for a reserve token at a given minimum price premium, emission rate, and price. These variables are set by Ohm controlled addresses, however a mechanism exists to issue more Ohm if a Chainlink oracle price of the reserve asset is sufficiently above the governance set minimum price premium.

A Reserve Migrator contract as also included in the review. The intent of this contract is to migrate DAI from DAI to USDS using a Maker provided conversion contract.

The contracts of the Olympus Emission Manager pull request were reviewed over 5 days. The code review was performed by 2 auditors between November 4, 2024 and November 9, 2024. The repository was under active development during the review, but the review was limited to the latest commit at the start of the review. This was commit e367e7977ea58a2fd365296d9c9f620c7cd0512d for the OlympusDAO/bophades repo.

Scope

The scope of the review consisted of the following contracts at the specific commit:

```
└── policies
    ├── EmissionManager.sol
    └── ReserveMigrator.sol
```

After the findings were presented to the Olympus team, fixes were made and included in several PRs.

This review is a code review to identify potential vulnerabilities in the code. The reviewers did not investigate security practices or operational security and assumed that privileged accounts could be trusted. The reviewers did not evaluate the security of the code relative to a standard or specification. The review may not have identified all potential attack vectors or areas of vulnerability.

yAudit and the auditors make no warranties regarding the security of the code and do not warrant that the code is free from defects. yAudit and the auditors do not represent nor imply to third parties that the code has been audited nor that the code is free from defects. By deploying or using the code, Olympus and users of the contracts agree to use the code at their own risk.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Good	All sensitive functions have appropriate access controls
Mathematics	Medium	There is frequent decimal conversion mathematics in the Emission Manager contract making the logic sometimes difficult to reason about.
Complexity	Medium	There are quite a few moving parts in the contract between the heart beats, optional rate changes, and variable updates depending on market conditions.
Libraries	Good	The contracts make use of Olympus tooling and standard libraries like Solmate.
Decentralization	Medium	The market is open, however keepers control the cadence of the system and protected users have the ability to set sensitive variables.
Code stability	Good	No changes were made to the PR after the audit began.
Documentation	Medium	No additional documentation was provided. However much of the code contains natspec and more complex functions contain useful comments, providing context.
Monitoring	Good	Nearly all functions emit events at the end of their execution.
Testing and verification	Good	Tests were extensive and thorough and include fuzz testing as well as unit testing.

Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
  - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas savings
  - Findings that can improve the gas efficiency of the contracts.
- Informational
  - Findings including recommendations and best practices.

Critical Findings

1. Critical - `EmissionManager::getSupply()` unexpectedly returns the supply in gOHM denomination

Technical Details

Currently, `EmissionManager::getSupply()` returns the total supply in gOHM denomination rather than OHM denomination, as the callers of `getSupply()` expect. This leads to a supply 10x larger than expected being emitted.

Impact

Critical

Recommendation

Divide `gohm.totalSupply() * gohm.index()` by `10 ** _gohmDecimals` rather than `10 ** _ohmDecimals`.

Developer Response

Fixed in commit `a2f6602a2913d683902a524b3e7e845509a277ad`

High Findings

None.

Medium Findings

1. Medium - Current market still active after `EmissionManager::shutdown()`

Technical Details

When a shutdown occurs, `locallyActive` is set to false, preventing keepers from creating new markets. However, this does not close the currently active market, which might be catastrophic depending on the nature of the emergency that required a shutdown.

Impact

Medium

Recommendation

Close the currently active market when `shutdown()` is called.

Developer Response

Fixed in commit `3ace544f24adfd3d218ae625b9d1449321f9e184`

Low Findings

1. Low - `backing` can inadvertently be set to 0

Technical Details

In `setBacking()` there is a check that `newBacking < (backing * 9) / 10`. However, there is no check that `newBacking != 0`. `backing` is used in division in various places in `EmissionManager`, so this value must never be 0.

Impact

Low

Recommendation

Check that `newBacking != 0` as is done in `initialize()`.

Developer Response

Fixed in commit `a1e708c5d4edf857b5d27db4ccb87d9c41721280`

2. Low - Unsafe `transfer()` and `approve()` used in `ReserveMigrator`

Technical Details

At the moment, the `ReserveMigrator` is intended to be used with DAI and USDS, so using `transfer()` and `approve()` will not revert or fail silently. However, this may not always be the case. `SafeTransfer()` and `SafeApprove()` should be used in place of `transfer()` and `approve()` to future-proof the contract for other assets.

Impact

Low

Recommendation

Use `SafeTransfer()` and `SafeApprove()` rather than `transfer()` and `approve()`.

Developer Response

Fixed in commit `35943667698bb8fc9290dadafa637f2e2905f338`

3. Low - No upper-bound is enforced for `restartTimeframe`

Technical Details

Currently, the `restartTimeframe` variable has no upper bound, meaning if it is set to a large value, the owner of the `emergency_restart` role can restart the contract without any governance feedback.

Impact

Informational

Recommendation

Ensure that `restartTimeframe` is <= the Olympus governance council proposal timeline, after which time governance could re-initialize the contract.

Developer Response

The intent is to set it to 11 days, which accomplishes this, but I'm not sure we need to hardcode the specific limit in the contract. The governance timeline can be changed so it

could get to a state where we weren't able to line them up.

Gas Saving Findings

1. Gas - Unnecessary OHM burn and reserve deposit in `EmissionManager::shutdown()`

Technical Details

The `EmissionManager` contract is never intended to hold OHM or the reserve asset therefore it should not have a balance to burn or deposit.

Impact

Gas Savings

Recommendation

Consider moving the fund recovery logic into a separate function such that `shutdown()` only controls the `locallyActive` variable.

Developer Response

Fixed in commit `6255ff18a5829f83ff20a35e9c46ed96b917f470`

Informational Findings

1. Informational - Unused Import in `EmissionManager`

Technical Details

`ReentrancyGuard` in `EmissionManager` is imported but unused and can be removed.

Impact

Informational

Recommendation

Removed the unused import

Developer Response

Fixed in commit `96f6a25d916ccae1510117c7a75cbc2c4f213838`

2. Informational - Incorrect variable name

Technical Details

The variable `beatsLeft` in `BaseRateChange` is incorrect as it is the number of days left in which to change the base rate, rather than the number of beats left.

Impact

Informational

Recommendation

Update the variable name such that it reflects the true nature of the variable.

Developer Response

Fixed in commit `fb90f53d52356e78775bab6c7de032cd34cc821d`

Final Remarks

Olympus uses the point-in-time price for the reserve asset, however, this is not susceptible to oracle manipulation due to the fact that the price is supplied by Chainlink and the keeper updates the price in the same transaction in which the price is used.

A gOHM flashloan attack to increase the supply and therefore the OHM emitted by the Emission Manager was discussed with the team. This was left out of the report because it was reported in a prior yAudit report and had already been acknowledged by the team.

In general, the code is extensively unit and fuzz tested. However, the auditors suggest that fork testing be done prior to deployment as it may have detected the sole critical finding in the report prior to the audit engagement.