# OlympusDAO

## Security Review

HickupHH3

7 November 2023

# Contents

# 1 Introduction

The purpose of this audit is to review 2 incremental changes to the Range-Bound Stability (RBS) mechanism:

1. Handling sDAI in TRSRY. Or in general, integration with ERC4626 tokens
2. Using asymmetric spreads + minting OHM tokens in `heart.beat()` + trigger a new rebase.

## 1.1 Audit Scope

The scope consisted of the `rbs-v1-4-asymmetric-with-sDAI` branch of the `bophades` repository at commit hash `a95bd5ee0c00f5b45392f1a7600e88a02932a11f`. The contracts found in the `src` folder that were included in scope were the following:

| File |
|---|
| interfaces/IStaking.sol |
| modules/RANGE/OlympusRange.sol |
| modules/RANGE/RANGE.v2.sol |
| policies/BondCallback.sol |
| policies/Distributor/Distributor.sol |
| policies/Distributor/ZeroDistributor.sol |
| policies/Heart.sol |
| policies/Operator.sol |
| policies/interfaces/IOperator.sol |
| policies/interfaces/IHeart.sol |
| policies/interfaces/IDistributor.sol |

## 1.2 Audit Timeline

The audit was conducted from **1st Nov** to **6th Nov**.

## 1.3 Fix Review

A review of the fixes was conducted subsequently on **7th Nov**.

## 1.4 Auditors Involved

HickupHH3

# 2 Risk Assessment Classification

There are 4 possible levels used to assess a vulnerability, with a separate section for gas optimizations.

## High

Directly exploitable vulnerabilities with medium / high likelihood of loss of user funds, or contract functionality.

Resolving these issues are crucial to ensure the security and functionality of the contracts.

## Medium

Vulnerabilities that relies on external dependencies / conditions to be met. Potentially leads to a loss of funds or functionality (eg. denial of service).

Resolving these issues are recommended to avoid undesired consequences.

## Low

Issues arising from deviant behaviour than expected, but has no / little bearing from a security standpoint.

## Informational

Issues that relate to security best practices recommendations, grammatical or styling errors, suggestions for variable/function name improvements etc. These issues are subjective and can be addressed based on the client's discretion.

While these issues may not directly affect the contract's functionality or security, addressing them can improve code readability, maintainability, and overall quality.

## Gas Optimizations

Suggested changes to the codebase that will help reduce deployment or runtime gas costs, or to reduce the bytecode size should the limit be reached.

# 3 Findings Summary

| Severity | No. of issues |
|---|---|
| High | 1 |
| Medium | 0 |
| Low | 0 |
| Informational | 2 |
| Gas Optimizations | 6 |
| Total | 9 |

## 3.1 [High] Incorrect amount withdrawn from `wrappedPayoutToken`

### Context

BondCallback.sol#L210

### Details

The amount withdrawn from `wrappedPayoutToken` uses the number of shares (`wrappedOutputAmount`) instead of the number of assets (`outputAmount_`).

There would fewer than expected assets withdrawn.

### Impact

The callback caller would receive less tokens than requested.

```
uint256 wrappedOutputAmount =
    wrappedPayoutToken.previewWithdraw(outputAmount_);
TRSRY.withdrawReserves(address(this), wrappedPayoutToken,
    wrappedOutputAmount);
wrappedPayoutToken.withdraw(wrappedOutputAmount, msg.sender, address(this));
```

### Mitigation

The correct implementation can be found in Operator.sol#L338-L345.

### Response

Fixed in PR#199.

### Status

Fixed. The implementation is consistent with `Operator.sol`.

## 3.2 [Info] Consider ensuring `reserve.decimals()` is equivalent to `wrappedReserve.decimals()`

### Context

Operator.sol#L69

Operator.sol#L112

### Details

There is a developer comment that states `_wrappedReserveDecimals == _reserveDecimals`, but this isn't checked on-chain.

### Mitigation

Consider checking that `_wrappedReserve.decimals()` is equal to `_reserveDecimals` in the constructor.

### Response

Using ERC4626 ensures that `reserve.decimals() == wrappedReserve.decimals()`, but we decided to perform the check just in case. Fixed in PR#199.

### Status

Fixed.

## 3.3 [Info] Use `IStaking.sol` instead of inline interfaces

### Context

IStaking.sol

Distributor.sol#L19-L26

ZeroDistributor.sol#L7-L14

### Details

The `Distributor` and `ZeroDistributor` defines a simplified `IStaking` interface with only the `unstake()` method, while there is a more complete `IStaking` interface that is used for tests.

### Mitigation

To avoid code duplication, consider defining the simple interface in `IStaking.sol`, then a more complete one with the additional methods that inherit it. The simple one can be imported into the distributor contracts.

### Response

Fixed in PR#199.

### Status

Fixed.

## 3.4 [Gas] Wall and spread checks can be reduced

### Context

OlympusRange.sol#L31-L38

OlympusRange.sol#L193-L195

### Details

The conditions checked are:

1. `ONE_PERCENT <= cushionSpread_`
2. `ONE_PERCENT <= wallSpread_`
3. `cushionSpread_ <= wallSpread_`

By checking (1) and (3), (2) is enforced. The same logic applies to the upper bound check for the low side.

### Mitigation

The referenced lines can be converted to the following:

```
if (
-   lowSpreads_[0] >= ONE_HUNDRED_PERCENT ||
    lowSpreads_[0] < ONE_PERCENT ||
    lowSpreads_[1] >= ONE_HUNDRED_PERCENT ||
-   lowSpreads_[1] < ONE_PERCENT ||
    lowSpreads_[0] > lowSpreads_[1] ||
    highSpreads_[0] < ONE_PERCENT ||
-   highSpreads_[1] < ONE_PERCENT ||
    highSpreads_[0] > highSpreads_[1] ||
) revert RANGE_InvalidParams();

if (
-   wallSpread_ < ONE_PERCENT ||
    cushionSpread_ < ONE_PERCENT ||
    cushionSpread_ > wallSpread_
) revert RANGE_InvalidParams();

- if (wallSpread_ >= ONE_HUNDRED_PERCENT || cushionSpread_ >=
    ONE_HUNDRED_PERCENT)
+ if (wallSpread_ >= ONE_HUNDRED_PERCENT)
    revert RANGE_InvalidParams();
```

### Response

Good catch, this will be useful as we have had to optimize some stuff in operator already due to codesize limits. Fixed in PR#199.

### Status

Fixed.

## 3.5 [Gas] Variables can be made immutable

### Context

Heart.sol#L54

ZeroDistributor.sol#L17

### Details

The referenced variables are either missing setters, or can be made immutable, depending on the intended behaviour.

### Mitigation

Either create setters for the referenced variables, or, for reduced gas costs, declare them as immutable.

### Response

Fixed in PR#199.

### Status

Fixed. `distributor` has a setter, while `staking` is made immutable.

## 3.6 [Gas] Equality case can be shifted

### Context

Heart.sol#L220

### Details

In the case where `currentTime - nextBeat == duration`, the calculated result will be `maxReward`. Hence, the return value can be directly assigned to `maxReward` like the strictly greater case.

### Mitigation

```
return
-     currentTime - nextBeat > duration
+     currentTime - nextBeat >= duration
       ? maxReward
       : (uint256(currentTime - nextBeat) * maxReward) / duration;
```

### Response

Fixed in PR#199.

### Status

Fixed.

## 3.7 [Gas] Cache `_config.regenObserve`

### Context

Operator.sol#L580

Operator.sol#L584

### Details

As `_config` is a storage variable, an extra SLOAD can be avoided by caching the value of `_config.regenObserve` and used in the referenced lines.

### Mitigation

```
+ uint32 regenObserve = _config.regenObserve;

- _updateRegenOnObservation(_status.low, currentPrice >= target,
    _config.regenObserve);
+ _updateRegenOnObservation(_status.low, currentPrice >= target,
    regenObserve);

- _updateRegenOnObservation(_status.high, currentPrice <= target,
    _config.regenObserve);
+ _updateRegenOnObservation(_status.high, currentPrice <= target,
    regenObserve);
```

### Response

Fixed in PR#199.

### Status

Fixed.

## 3.8 [Gas] Pre-in/decrements cost less gas compared to post-in/decrements

### Context

Operator.sol#L596

Operator.sol#L601

### Mitigation

```
- regen_.count++;
+ ++regen_.count;
```

```
- regen_.count--;
+ --regen_.count;
```

#### Response

Fixed in PR#199.

#### Status

Fixed.

### 3.9 [Gas] `_regenerate()` can be unchecked

#### Context

Operator.sol#L609-L666

#### Details

Relevant checks such that the subtractions in the `_regenerate()` function will not overflow. Hence, the function can be wrapped in an unchecked block for greater gas efficiency.

#### Mitigation

Wrap the function in an unchecked block.

#### Response

Fixed in PR#199.

**Status**

Fixed.

# 4 Disclaimer

The audit report provided reflects a thorough review conducted to the best of my ability. However, it is important to note that the time-boxing nature of the review and available resources may prevent the discovery of all potential security vulnerabilities. As such, this audit does not guarantee the absence of undiscovered vulnerabilities.

Furthermore, please be aware that the security review was conducted on a specific commit of the codebase, as indicated. Any subsequent modifications made to the code will necessitate a new security review to ensure comprehensive coverage.

Note that the contracts used in production and expected deployment values may defer significantly from what was reviewed.

To ensure a robust evaluation of the codebase, it is highly recommended to engage multiple auditors and firms, particularly for large and complex projects. The involvement of multiple perspectives can provide additional insights and potential missed vulnerabilities.

Please consider these factors when assessing the audit report and making decisions related to the security and reliability of the smart contracts. The security review is not an endorsement of the project or its team, and should not be treated as such.