OlympusDAO

Security Review

HickupHH3

4 November 2024

Contents

1 Introduction

The purpose of this audit is to review the Cooler Loans LoanConsolidator contract and associated contracts.

1.1 Audit Scope

The scope consisted of the `coolerutils-improvements` branch of the `bophades` repository initially at commit hash `b6bb5075843a6f6b703b740d0a266e831ba9b8fe`. Subsequently, some changes were made in PR 432, namely:

- Enabling ownership change while consolidating
- Allow consolidation of a single loan (instead of the previous minimum of 2) when consolidating across different Coolers
- Disabling contract functionality by default

The commit hash with these changes is

`95479d5d4a9bb941c60c7a8347709d9fc895b819`.

The contracts found in the `src` folder that were included in scope were the following:

File
interfaces/maker-dao/IERC3156FlashBorrower.sol
interfaces/maker-dao/IERC3156FlashLender.sol
modules/CHREG/CHREG.v1.sol
modules/CHREG/OlympusClearinghouseRegistry.sol
modules/RGSTY/RGSTY.v1.sol
modules/RGSTY/OlympusContractRegistry.sol
policies/ContractRegistryAdmin.sol
policies/LoanConsolidator.sol

1.2 Audit Timeline

The audit was conducted from **28th Oct** to **31st Oct**.

1.3 Fix Review

A review of the fixes was conducted subsequently on **1st Nov** to **4th 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	0
Medium	0
Low	3
Informational	2
Gas Optimizations	3
Total	8

3.1 [Low] Incorrect `clearinghouse` used for collateral calculation

Context

LoanConsolidator.sol#L464-L466

Details

`clearinghouseFrom` is used to calculate the collateral required for the consolidated loan, but it should be `clearinghouseTo`. This will pose problems if the clearing houses have different LTVs, specifically if `clearinghouseFrom` calculates a lower collateral amount, as it'll cause a revert for the request principal in `clearinghouseTo`.

Recommendation

```
- uint256 consolidatedLoanCollateral =
    flashLoanData.clearinghouseFrom.getCollateralForLoan(
+ uint256 consolidatedLoanCollateral =
    flashLoanData.clearinghouseTo.getCollateralForLoan(
    flashLoanData.principal
);
```

Response

Fixed in 1ffea94.

Status

Fixed.

3.2 [Low] `lenderFee` should be exclusive of `totalInterest`

Context

LoanConsolidator.sol#L809

LoanConsolidator.sol#L901

Details

The `totalInterest` is expected to be paid by the caller, not from the flash loan, where `flashloanAmount = totalPrincipal;`. Hence, the `lenderFee` calculated should not include `totalInterest`.

Recommendation

```
- uint256 lenderFee = FLASH.flashFee(address(DAI), totalPrincipal +
    totalInterest);
+ uint256 lenderFee = FLASH.flashFee(address(DAI), totalPrincipal);
```

Response

Fixed in f70896c.

Status

Fixed.

3.3 [Low] Contract names can be visually identical but have different ascii representations

Context

OlympusContractRegistry.sol#L231-L232

Details

The contract name validation accepts null characters. Hence, it is possible to have visually identical contract names with different `byte5` representations. For example, `abab` can be represented as `0x6162006162` or `0x0061626162`.

Recommendation

The desired behaviour would be to allow only null characters at the tail end of the contract name.

```
if (char == 0x00) {
    for (uint256 j = i; j < 5; j++) {
        if (char != 0x00) revert Params_InvalidName();
    }
}
```

Response

Fixed in 31732c7 and fc92f01.

Status

Fixed.

3.4 [Info] Comment / minor code improvements

Context

CHREG.v1.sol#L4

CHREG.v1.sol#L9

OlympusClearinghouseRegistry.sol#L10

CHREG.v1.sol#L50

LoanConsolidator.sol#L441

LoanConsolidator.sol#L458-L461

LoanConsolidator.sol#L487

LoanConsolidator.sol#L496

LoanConsolidator.sol#L791

OlympusContractRegistry.sol#L216-L218

LoanConsolidator.sol#L240

LoanConsolidator.sol#L288

Details

The referenced lines are either comments to be edited for clarity or code to be modified for consistency.

Recommendation

```diff
- import "src/Kernel.sol";
+ import {Module} from "src/Kernel.sol";

- single-soure of truth
+ single-source of truth

- clearginhouse
+ clearinghouse

- // Fees are in terms of the reserveTo token
+ // Interest was collected in terms of the reserveTo token

// State:
// - reserveFrom: reduced by principal and interest, should be 0
- // - reserveTo: no change, should be 0
+ // - reserveTo: lender fee + protocol fee
// - gOHM: increased by the collateral returned
```

```
- // Transfer the amount of `reserveTo` required to repay the flash loan
    (debt + interest), lender fee and protocol fee
+ Transfer the principal amount of `reserveTo`. The lender fee and protocol
    fee have already been transferred.

// State:
// - reserveFrom: no change
- // - reserveTo: increased by the flashloan amount, lender fee and protocol
    fee
+ // - reserveTo: flash loan principal + lender fee + protocol fee
// - gOHM: no change, 0

- /// @return ownerGOhm           Amount of gOHM to be approved by the Cooler
    owner.
+ /// @return GOhmAmount          Amount of gOHM to be approved by the Cooler
    owner.

/// @notice Validates the contract name
/// @dev    This function will revert if:
- ///         - The name is empty

- ///         - The caller has not approved this contract to spend the fees
    in DAI.
- ///         - The caller has not approved this contract to spend the
    reserve token of `clearinghouseTo_` in order to repay the flashloan.
+ ///         - The caller has not approved this contract to spend the
    reserve token of `clearinghouseTo_` in order to pay the interest, lender
    and protocol fees
```

Response

Fixed in 4aa38c5.

Status

Fixed.

3.5 [Info] Redundant `sDAI` token variable

Context

LoanConsolidator.sol#L122-L124

Details

`sDAI` is defined and fetched from the registry, but the consolidator no longer supports `sDAI` payments.

Recommendation

Remove the `SDAI` variable.

Response

Fixed in a641c10.

Status

Fixed.

3.6 [Gas] Shift `GOHM.transferFrom()` after collateral calculation

Context

LoanConsolidator.sol#L470-L476

LoanConsolidator.sol#L624-L626

Recommendation

The `gOHM` collateral transfer from the cooler owner can be performed after determining the required `consolidatedLoanCollateral`, so that just 1 transfer is required.

Response

Fixed in 1ffea94 and b4951b8.

Status

Fixed.

3.7 [Gas] New cooler owner verification can be optimised

Context

LoanConsolidator.sol#L314-L318

Details

- `Cooler(coolerFrom_).owner()` is checked to be `msg.sender` prior to the different owner check, so it can be replaced to avoid an external call.
- Because the `coolerFrom` and `coolerTo` owners (which are immutable) must be different, it implicitly ensures that the coolers are different, making the explicit check redundant.

Recommendation

```
- // Ensure that the caller is not trying to operate on the same Cooler
- if (coolerFrom_ == coolerTo_) revert Params_InvalidCooler();

// Ensure that the owner of the coolerFrom_ is not the same as coolerTo_
```

```
+ // This also implicitly checks that the coolers must be different, ie.
    can't operate on the same Cooler
- if (Cooler(coolerFrom_).owner() == Cooler(coolerTo_).owner()) revert
    Params_InvalidCooler();
+ if (msg.sender == Cooler(coolerTo_).owner()) revert Params_InvalidCooler();
```

Response

Fixed in 1ffea94.

Status

Fixed.

3.8 [Gas] Iterating through `contractNames` is redundant

Context

OlympusContractRegistry.sol#L242-L248

OlympusContractRegistry.sol#L256-L262

Details

When `_updateImmutableContractNames()` / `_updateContractNames()` is invoked, `name_` is guaranteed to not be in `_immutableContractNames` / `_contractNames`. This is because in the former case, `_name` can only be mapped once, and thus cannot be reused to register another contract. In the latter, the name will be popped upon de-registration of the contract.

Recommendation

`_updateImmutableContractNames()` and `_updateContractNames()` can actually be removed, and be inlined in the functions `registerImmutableContract()` / `registerContract()` where they're invoked.

```
_immutableContracts[name_] = contractAddress_;
- _updateImmutableContractNames(name_);
+ _immutableContractNames.push(name_);

_contracts[name_] = contractAddress_;
- _updateContractNames(name_);
+ _contractNames.push(name_);
```

Response

Fixed in 57ab1f3.

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.