PUBLIC

Code Assessment

of the D3M ERC-4626 Smart Contracts

March 21, 2024

Produced for

by

Contents

1 Executive Summary 3
2 Assessment Overview 5
3 Limitations and use of report 7
4 Terminology 8
5 Findings 9
6 Informational 10
7 Notes 11

1 Executive Summary

Dear all,

Thank you for trusting us to help MakerDAO with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of D3M ERC-4626 according to Scope to support you in forming an opinion on their security risks.

MakerDAO has implemented new components for the existing D3M v2 system: an ERC-4626 compatible pool designed for use with MetaMorpho and a plan that enables an operator to set a target asset amount.

The most critical subjects covered in our audit are asset solvency, functional correctness and the correct integration into the existing D3M v2 system.

The general subjects covered include compliance with ERC standards and maintaining the consistency of the codebase.

In summary, we find that the codebase provides a high level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	1
• Risk Accepted	1

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

The assessment was performed on the source code files inside the D3M ERC-4626 repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	17 March 2024	698ec5c43bb76f3ccf850d229561e3e900e9c2b8	Initial Version

For the solidity smart contracts, the compiler version `0.8.14` was chosen. The following files were in scope:

```
src/plans/D3MOperatorPlan.sol
src/pools/D3M4626TypePool.sol
```

2.1.1 Excluded from scope

All other files and the correctness of the external systems are out of scope.

2.2 System Overview

This system overview describes the initially received version (Version 1) of the contracts as defined in the Assessment Overview.

Furthermore, in the findings section, we have added a version icon to each of the findings to increase the readability of the report.

MakerDAO implements a D3M pool for tokenized vaults (ERC-4626), with the initial use case of depositing into Metamorpho, along with a D3M plan that allows an operator to set the target amount for the allocated DAI.

For details regarding the D3M architecture and implementation details, see our D3M v2 report.

2.2.1 D3M4626TypePool

This section iterates over the implementation of the ERC-4626 D3M pool functions. The D3M v2 report highlights their usage in the context of the D3M hub.

The `D3M4626TypePool` supports depositing DAI into tokenized vaults. Namely, the functions `deposit()` and `withdraw()` simply call the ERC-4626's `deposit()` and `withdraw()` functions respectively. As for other D3M pools, the shares are held in the pool while the redeemed DAI is sent to the D3M hub to be handled.

The `exit()` function allows the hub to distribute the remaining pool shares if the shutdown has been activated. Similar to other pools, the `D3M4626TypePool` distributes the shares proportionally (to gem balances of users).

The `quit()` method allows authorized addresses to send the held shares to arbitrary addresses. The assumption that this is only possible when no shutdown has occurred (as outlined in the original report) holds for the novel pool type.

The hooks `preDebtChange()` and `postDebtChange()` both remain empty. It is assumed that the ERC-4626 does not require any actions before any interaction. Further, it is assumed that the pool's getter functions (integration with ERC-4626, see below) do not require any actions to return the most recent values.

The D3M pool's getter function `assetBalance()` computes the managed DAI balance by leveraging the ERC-4626's `convertToAssets()` function while the `maxDeposit()` and `maxWithdraw()` getters leverage the corresponding functions in the ERC-4626 to return the maximum deposit and withdraw respectively.

Like other pools, the `D3M4626TypePool` has the standard `rely()` and `deny()` authorization functions as well as a `file()` function to set the D3M hub.

2.2.2 D3MOperatorPlan

This section elaborates on the implementation of the operator plan. The D3M v3 report puts the usage of the function into perspective.

The introduced `D3MOperatorPlan` allows authorized addresses to choose an operator that sets the target asses amount. Hence, `getTargetAssets()` corresponds to the amount set by the operator (if enabled, otherwise it is zero). Like other plans, the plan can be deactivated through `disable()` so that `active()` returns `false` (only condition).

Like other plans, the `D3MOperatorPlan` has the standard `rely()` and `deny()` authorization functions as well as `file()` functions to set the operator or to disable or enable the plan.

2.2.3 Trust Model & Roles

The ERC4626 pool inherits the trust model of the D3M core and trusts the D3M hub to be behaving as outlined in the report. It is also expected that the supported ERC4626 tokens (e.g. Metamorpho) do not make a loss as of the trust model section in the report.

The tokenized vaults such as Metamorpho are central to the functionality of the D3M. For example, a Metamorpho vault could allocate to malicious vaults. Thus, the external system is trusted not to misbehave.

Further, note that the ERC-4626 tokens are standard ones and do not have any specific behaviour (e.g. disabled functions, additional steps required).

The ``auth``ed addresses in the pool are fully trusted since they could transfer out all held shares to arbitrary addresses.

The ``auth``ed addresses are expected to set a reasonable operator while the operator is expected to set a reasonable target amount (note that the D3M limits the target amount additionally by limiting the maximum debt for a pool).

3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

4 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- *Likelihood* represents the likelihood of a finding to be triggered or exploited in practice
- *Impact* specifies the technical and business-related consequences of a finding
- *Severity* is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

Likelihood	Impact		
	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

5 Findings

In this section, we describe our findings. The findings are split into these different categories:

- Correctness: Mismatches between specification and implementation

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	1

- Temporary DoS Due to Max Deposit in Metamorpho Risk Accepted

5.1 Temporary DoS Due to Max Deposit in Metamorpho

Correctness Low Version 1 Risk Accepted

CS-D3M4626-001

The Metamorpho contract's `maxDeposit()` function may overestimate the maximum amount of assets that can be deposited:

```
/// @dev Warning: May be higher than the actual max deposit due to duplicate markets
in the supplyQueue.
```

Hence, it could be possible, that the D3M may try to deposit more than possible into the pool. Ultimately, the D3M could revert and deposits could be temporarily DoSed.

Note that for regular ERC-4626 tokens, the following holds:

```
MUST return the maximum amount of assets deposit would allow to be deposited for
receiver and not cause a revert, which MUST NOT be higher than the actual maximum
that would be accepted (it should underestimate if necessary).
```

To summarize, deposits could be theoretically DoSed due to an overestimation of the maximum allowed deposit for Metamorpho.

Risk accepted:

MakerDAO is aware of the issue.

6 Informational

We utilize this section to point out informational findings that are less severe than issues. These informational issues allow us to point out more theoretical findings. Their explanation hopefully improves the overall understanding of the project's security. Furthermore, we point out findings which are unrelated to security.

6.1 Lack of NatSpec and Documentation

Informational **Version 1**

CS-D3M4626-002

While the README and comments generally document the codebase sufficiently, adding further documentation could be helpful. Namely, the following parts could be improved:

1. The `D3MOperatorPlan`, in contrast to the new ERC4626 pool, does not have any NatSpec (inherited from its interface).
2. The README contains descriptions for other pools and plans. However, it has not been extended to include the new contracts.

7 Notes

We leverage this section to highlight further findings that are not necessarily issues. The mentioned topics serve to clarify or support the report, but do not require an immediate modification inside the project. Instead, they should raise awareness in order to improve the overall understanding.

7.1 Imprecise Handling of Standard ERC-4626 With Fees

Note Version 1

EIP-4626 defines the `convertToAssets()` to satisfy the following property:

```
MUST NOT be inclusive of any fees that are charged against assets in the Vault.
```

While Metamorpho provides an accurate estimate for mints and burns, the standard specifies that these should not be included. Hence, for tokenized vaults with fees charged against assets in the vault (e.g. withdrawal fees), `convertToAssets()` may overestimate the assets that will be received (for ERC-4626 with fees).

Note that `convertToAssets()` does not include other parameters in its calculation (e.g. slippage) and hence the number of assets managed could be underestimated in certain scenarios.

As a consequence, the D3MHub may make a wrong decision regarding winding or unwinding or an imprecise one.