PUBLIC

Code Assessment

of the D3M AaveV3 USDS Pool Smart Contracts

September 10, 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 Resolved Findings 10

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 AaveV3 USDS Pool according to Scope to support you in forming an opinion on their security risks.

MakerDAO has implemented a new component for the existing D3M v2 system: a pool supporting USDS deposits into Aave V3-like protocols without supply caps, such as SparkLend.

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 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	0

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 AaveV3 USDS Pool 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	29 Aug 2024	758b26cdd29b5be0a71a98d76b5e53be5c97b63c	Initial Version
2	09 Sep 2024	f2b764f4c7950217a22762a1bf81ecedd7a9a3b7	After Intermediate Report

For the solidity smart contracts, the compiler version `0.8.21` was chosen. The following files were in scope of this review:

```
./src/pools/D3MAaveV3USDSNoSupplyCapTypePool.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 depositing USDS into Aave V3-like protocols without supply caps, such as SparkLend. While remaining similiar to the DAI pool, the new USDS pool additionally converts the DAI to USDS. For details regarding the D3M architecture and implementation, see our D3M V2 report.

2.2.1 AaveV3 USDS No Supply Cap Pool

This section iterates over the implementation of the `D3MAaveV3USDSNoSupplyCapTypePool` functions. The D3M V2 audit report highlights their usage in the context of the D3M hub.

The `D3MAaveV3USDSNoSupplyCapTypePool` supports depositing USDS into Aave V3-like protocols. While `deposit()` and `withdraw()` perform regular deposits and withdrawals, they additionally convert DAI to USDS and vice-versa, respectively.

`exit()` distributes, similar to other pools, the held shares (aUSDS) proportionally (to gem balances of users), in case the shutdown has been activated. The `quit()` function allows authorized addresses to

send the held aUSDS 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. The D3M pool's getter function `assetBalance()` returns the aUSDS balance (since USDS is 1:1 convertible to DAI). The `maxDeposit()` returns `uint256.max` indicating that no pools with supply caps are supported while `maxWithdraw()` returns the minimum of the aUSDS balance held (claimable amount) and the lending pool's USDS balance (currently available amount).

Like other pools, the `D3MAaveV3USDSNoSupplyCapTypePool` has the standard `rely()` and `deny()` authorization functions as well as a `file()` function to set the D3M hub or the king. Similar, to the other Aave V3 D3M pool's, a `collect()` function is implemented for claiming rewards and sending them to `king`.

2.2.2 Trust Model & Roles

Wards: Each address set to 1 in the wards mappings is fully trusted and expected to behave correctly.

Aave V3: Aave V3's contracts are central to the functionality of the D3M and are upgradeable. Hence, Aave V3 is fully trusted to not make any malicious modifications to the contracts.

MakerDAO assumes that both protocols return a positive interest rate such that `assetBalance() >= ink`.

In the case where Aave V3 faces a problem that would trigger an asset loss, it could break the latter assumption. Note that the debt generated for each ilk is only protected by an upward debt ceiling. This means that the loss case is not handled but is only partially mitigated.

Please consider the general trust model for the D3M as outlined in the initial D3M report.

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 any open findings. Findings that have been resolved have been moved to the Resolved Findings section. The findings are split into these different categories:

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	0

6 Resolved Findings

Here, we list findings that have been resolved during the course of the engagement. Their categories are explained in the Findings section.

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	0
Informational Findings	1

- Wrong Error Message Code Corrected

6.1 Wrong Error Message

Informational Version 1 Code Corrected

CS-MKD3MUSDS-001

The error messages of the D3M contracts are typically encoded as `<contract name>/<error message>`. However, the `D3MAaveV3USDSNoSupplyCapTypePool` reverts with the messages `D3MAaveV3NoSupplyCapTypePool/<error message>`.

Code corrected:

Now the messages are encoded as `D3MAaveV3USDSNoSupplyCapTypePool/<error message>`.