PUBLIC

Code Assessment

of the DAI Rates & sDAI Oracle Smart Contracts

April 26, 2023

Produced for

by

Contents

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 DAI Rates & sDAI Oracle according to Scope to support you in forming an opinion on their security risks.

MakerDAO implements an interest rate strategy for DAI for the Spark Lend protocol and a price oracle for SavingsDAI, an ERC-20 representation of Pot positions.

The most critical subjects covered in our audit are functional correctness and precision of arithmetic operations. Security regarding all the aforementioned subjects is high.

The general subjects covered are specification and documentation.

In summary, we find that the codebase provides a good 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 DAI Rates & sDAI Oracle 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	09 February 2023	ea3f2e1262a1eed8ab702473e3d7121a38dccb05	Initial Version
2	19 April 2023	8a2a04b0b708667326b6a169a072da6115e2638c	Second Version
3	24 April 2023	a20c60f83b8b35b18dd3bcba10678f7b5b3005b8	Third Version

For the solidity smart contracts, the compiler version `0.8.10` was chosen.

2.1.1 Excluded from scope

Anything besides `DaiInterestRateStrategy.sol` and `SavingsDaiOracle.sol` is out of scope. The contracts using these are expected to use these contracts correctly.

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 offers an interest rate strategy and a price oracle for savings DAI that is compatible with Spark Lend. Savings DAI is a tokenized representation of positions on MakerDAO's Pot contract which allows users to deposit their DAI to earn interest according to the Dai Savings Rate. Further, an interest rate strategy for DAI for Spark Lend is implemented that offers borrow rates around the current DAI Savings Rate under regular conditions.

2.2.1 Price Oracle

The price oracle computes the price of savings DAI shares. Namely, it implements following functionality from Aave V3's AggregatorInterface:

- `latestAnswer()`: Computes the latest price by querying the DAI price feed's latest answer and multiplying it with `chi`, the rate accumulator in the Pot contract that decides how much a share is worth in DAI.
- `latestTimestamp()`: Returns the latest round's timestamp of the DAI oracle.
- `latestRound()`: Returns the latest round's ID of the DAI oracle.
- `getAnswer()`: Performs the same computation as `latestAnswer()` on a given round id using the DAI oracle's `getAnswer()` function.

- `getTimestamp()`: Gets the timestamp of a given round id.

Other functions are getters for the DAI price feed and MakerDAO's Pot contract.

2.2.2 Interest Rate Strategy

The interest rate strategy is intended to be used by Spark Lend pool that is supplied by a D3M implementation. Further, is implemented for DAI so that a D3M implementation for the Spark Lend protocol should be able to unwind as fast as possible in case of debt limit changes downwards by incentivizing borrowers and lenders to move DAI into the protocol. Hence, it operates in two modes. Namely, it distinguishes the unhealthy scenario, where the D3M supplied too much (the Spark Lend D3M ink's debt exceeds the debt limit), from the healthy one, where the D3M is healthy.

Note that the maximum rate and the borrow and supply spreads are constants, while the DSR rate is queried from Maker's Pot contract. The base rate is defined as

$$r_{base} = min(r_{dsr}, r_{max} - r_{borrowSpread})$$

Meaning, that the sum of the base rate and the borrow spread cannot exceed the maximum rate.

Assume the D3M is healthy. The borrow rate is a constant defined as the Dai Savings Rate plus a borrow spread. While the borrowed amount is below a certain performance value, the supply rate is set to zero. Once the borrowed amount reaches the performance value, the supply rate is computed as the Dai Savings Rate plus a supply spread, multiplied with the ratio of the amount, that went over the premium and the total liquidity (borrows + available capital) in the pool. Hence, the rates can be described as follows:

$$r_{borrow} = r_{base} + r_{borrowSpread}$$
$$r_{supply} = (r_{base} + r_{supplySpread})\frac{max(0, c_{borrowed} - c_{performance})}{c_{borrowed} + c_{available}}$$

Note it yields that the borrow rate is always constant. Further, suppliers are only incentivized to supply capital after a minimum borrow amount is reached. During the times, as the third-party suppliers are not incentivized, we expect that the D3M provides sufficient DAI for the lending market. However, once the protocol makes sufficient profits, it will incentivize third party suppliers (as the D3M will have a certain debt limit).

In case the D3M is unhealthy, meaning that the debt is higher than the debt limit, the D3M will try to wind down. In that scenario, the interest rate strategy will try to incentivize borrowers to pay back their debt, and will try to incentivize suppliers to start lending DAI. Hence, the borrow and supply rate will increase according to the debt ratio of the D3M. More specifically, the rates are defined as

$$r_{borrow} = r_{max} - \frac{r_{max} - (r_{base} + r_{borrowSpread})}{debtRatio}$$
$$r_{supply} = \frac{c_{borrowed}}{c_{borrowed} + c_{available}} r_{borrow}$$

Note, that if the debt ratio increases, the borrow rate increases as a negated inverse function (starting at the regular borrowing rate). Similarly, the supply rate will increase in terms of the debt ratio, however, scaled by the utilization ratio. Thus, the higher the utilization, the closer will the supply rate be to the borrow rate. Ultimately, that leads to the protocol forfeiting potential revenue by sharing it with third-party supplier to incentivize the D3M's stabilization. Note that the stable borrow rate is always 0.

The interest rate definition described above is implemented in `calculateInterestRates()`. However, note that the debt ratio and the base rate are both not queried on every interest rate calculation; but, retrieved from a cache (as these will not change often). The variables can be recomputed with function `recompute()` that sets the base rate to the current Dai Savings Rate and computes the debt ratio as the ratio of the current `Ilk.Art` and current `Ilk.line`. It is assumed that the recomputation is triggered on a regular basis.

2.3 Changes in Version 2

The stable fee base rate (SFBR) is now used instead of the DSR. Note that it is assumed that there is a conversion factor from DSR to SFBR so that

$$r_{sfbr} = b * r_{dsr}$$

holds.

Further, the assumption that the `ilks` rate is constant has been lifted. We assume that the conversion factor is a constant.

2.4 Changes in Version 3

Now, in case of a shutdown of the Maker system, the rate will stay constant at the maximum rate.

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:

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 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.

6.1 Timestamps and Rounds

Note Version 1

The timestamps and round ids returned by the DAI savings oracle are equivalent to the ones of the DAI oracle. However, it could be that the `drip()` function on the Pot has not been called for a long time. Hence, there could be a high mismatch between the freshness timestamp and the actual freshness.