PUBLIC

Code Assessment

of the Interest Rate Strategy Smart Contracts

July 27, 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 Interest Rate Strategy according to Scope to support you in forming an opinion on their security risks.

MakerDAO implements a new interest rate strategy for the Aave v3 fork Spark Lend that sets the interest rate for the Spark Lend DAI market to the base DAI savings rate. In comparison to the old version, the contract retrieves the base rate from the "ETH-C" ilk (collateral type) of the Maker contract `Jug` instead of the DSR rate from the Maker contract `Pot`.

The most critical subjects covered in our audit are functional correctness and the correct adherence to the MakerDAO specifications. We have high confidence on both subjects although a certain `base` variable is omitted where no official specification indicates that it is not in use.

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 Interest Rate Strategy repository based on the documentation files. The scope is limited to the contract `DaiJugInterestRateStrategy.sol`.

The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	24 July 2023	3d5d8b5ff2a834b4cd00ad2e1c8dfc74f093215a	Initial Version

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

2.1.1 Excluded from scope

Any other file not explicitly mentioned in the scope section. In particular tests, scripts, external dependencies, and configuration files are not part of the audit 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 offers an interest rate strategy for Aave v3 which is used in conjunction with a custom price oracle and a Maker Direct Deposit Module (D3M) to create a market for the DAI stable-coin in Spark Lend which is an Aave v3 fork that centers on DAI. Interest rates are based on the DAI savings rate (DSR) - the base rate that is awarded to DAI stakers through the Savings DAI contract. Rates are kept flat by the associated D3M that maintains a buffer of DAI liquidity on the protocol.

2.2.1 DaiJugInterestRateStrategy

The `DaiJugInterestRateSTtrategy` implements Aave v3's `IReserveInterestRateStrategy` interface and exposes the function `calculateInterestRates()` which is used in Aave v3's `ReserveLogic.updateInterestRates` to set the interest rates for supplying as well as borrowing (stable and dynamic). As stable borrowing rates are not used by the protocol, the contract always sets this rate to 0. Supply rates and variable borrow rates are based on a `debtRatio` value that has to be updated occasionally by calling the `recompute()` of the contract.

The strategy has its own ilk (Maker collateral type) associated to it, which defines a maximum debt ceiling: "DIRECT-SPARK-DAI". As the D3M tries to allocate further funds to the protocol to maintain the given buffer, it might hit this debt ceiling in which case no more additional liquidity can be provided. In this case, the aforementioned `debtRatio` comes into play. It is calculated by dividing the current debt of the protocol by the debt ceiling. Values larger than 1 (RAY) indicate that the debt ceiling has been hit and the buffer might shrink. In that case, borrowing is discouraged to decrease utilization.

Depending in the `debtRatio`, interest rates are computed two-fold:

1. If the `debtRatio` is smaller or equal to 1:
    - The borrow rate is equal to the DSR.
    - Outstanding borrows from 0 to a `performanceBonus` threshold accrue no supply interest. From the threshold onwards, the supply interest is the equal to the DSR times the current utilization of the market.
2. If the `debtRatio` is greater than 1:
    - The borrow rate can approach a maximum value (currently 75%) depending on the debt ratio.
    - The supply rate is equal to the borrow rate times the current utilization of the market.

In the second case, borrow rates increase fast to encourage borrowers to return their borrowed DAI and, if utilization is high, to encourage third party lenders to supply additional DAI to the market.

Usually, the supply rate is the rate that earns interest for the Maker DAO as most DAI tokens will be supplied by the D3M and not third party suppliers. The `performanceBonus` threshold allows the SubDAO that operates the Spark Lend protocol to earn these fees up to the threshold instead (by setting the supply rate to 0).

Both supply and borrow rate can be increased with a static spread that is determined at deployment and increases earnings for MakerDAO or the SubDAO, depending on how they are set up.

It is worth to note that the interest rates are based on the base DSR (stability fee contribution of the "ETH-C" ilk in `Jug` of Maker Core). In certain circumstances, the real DSR for Savings DAI (`Pot.dsr()` in Maker Core) can be higher.

The stability fee contribution of an ilk in Maker Core is calculated by the `duty` variable of an ilk and additionally by a `base` variable in the `Jug` contract. The `base` variable is not accounted for in the implementation of the `DaiInterestRateStrategy`. MakerDAO has assured that this variable is not in use and will never be used although no formal specification exists that indicates this behavior.

2.2.2 Roles & Trust Model

Apart from `debtRatio` and `baseRate`, there are no mutable storage variables in the contract. Both values are set by a public function that can be executed by anyone. They are based on values that are set in the Maker Core contracts which are governed by the MakerDAO.

The contract is immutable.

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