MakerDAO: Rates Conv

Security Review

Cantina Managed review by:
Chris Smith, Lead Security Researcher
Xmxanuel, Lead Security Researcher

September 4, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

The Maker Protocol, also known as the Multi-Collateral Dai (MCD) system, allows users to generate Dai (a decentralized, unbiased, collateral-backed cryptocurrency soft-pegged to the US Dollar) by leveraging collateral assets approved by the Maker Governance, which is the community organized and operated process of managing the various aspects of the Maker Protocol.

From Mar 10th to Mar 21st the Cantina team conducted a review of rates-conv on commit hash af1fc165. The team identified **1** issue:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	1	1	0
Gas Optimizations	0	0	0
Informational	0	0	0
Total	**1**	**1**	**0**

The Cantina Managed team reviewed MakerDAO's `rates-conv` holistically on commit hash 26921f3e and concluded that all issues were addressed and no new vulnerabilities were identified.

3 Findings

3.1 Low Risk

3.1.1 Incorrect `dataSlot` computation in `btor()` function

Severity: Low Risk

Context: Conv.sol

This issue was independently identified and reported by MakerDAO.

Description: The `btor()` function contains a bug in the computation of the data slot for the `RATES` mapping. The original implementation incorrectly uses the free memory pointer area for computing the keccak256 hash, which could cause function calls to fail if the `RATES` storage slot is not 0.

The issue arises because the assembly block was not using memory-safe practices and was computing the hash without properly setting up the memory layout. This could lead to unpredictable behavior when the storage slot value is non-zero. Although it does not affect the deployed contract (the storage layout cannot change after deployment), future deployments should favor the fixed version.

```
(uint256 wordPos, uint256 bytePos) = (offset / 32, offset % 32);

bytes32 value;
assembly {
    let dataSlot := keccak256(RATES.slot, 0x20)  // Incorrect: uses arbitrary memory
    value := sload(add(dataSlot, wordPos))
}
```

Recommendation: Use proper scratch space for memory operations in assembly blocks and ensure memory-safe practices are followed. The data slot should be computed by first storing the slot value in a known memory location before hashing.

MakerDAO: Fixed in commit 26921f3e.

Cantina Managed: Fix verified.