CANTINA

MakerDAO: Spark Rewards

Security Review

Cantina Managed review by:
Christoph Michel, Lead Security Researcher
M4rio.eth, Security Researcher

February 27, 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 Feb 18th to Feb 19th the Cantina team conducted a review of spark-rewards on commit hash 00478a8c. The directories in scope for this review were:

- `src`
- `script`

The team identified a total of **2** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 0
- Gas Optimizations: 1
- Informational: 1

The Cantina team reviewed MakerDAO's holistically on tag v1.0.0 and concluded that all issues were addressed and no new vulnerabilities were identified.

3 Findings

3.1 Gas Optimization

3.1.1 Consider using `MerkleProof.verifyCalldata` version to consume fewer gas

Severity: Gas Optimization

Context: SparkRewards.sol#L98

Description: OpenZeppelin's `MerkleProof` library also offers a `MerkleProof.verifyCalldata` function that can be used if the Merkle proof is a `calldata bytes` array. It's more gas efficient than the `MerkleProof.verify` function as it does not copy the Merkle proof calldata to memory.

Recommendation: Consider using the `MerkleProof .verifyCalldata` version.

```
- require(MerkleProof.verify(merkleProof, expectedMerkleRoot, leaf), "SparkRewards/invalid-proof");
+ require(MerkleProof.verifyCalldata(merkleProof, expectedMerkleRoot, leaf), "SparkRewards/invalid-proof");
```

MakerDAO: Fixed in PR 9.

Cantina Managed: Verified.

3.2 Informational

3.2.1 Document required Merkle tree properties for `spark-rewards`

Severity: Informational

Context: README.md

Description: For the spark-rewards cumulative claims to function properly, the Merkle tree is expected to adhere to the following properties:

1. The Merkle tree can contain several epochs.
2. The Merkle tree's leaves are not removed unless the leaves' epoch is permanently closed.
3. When new rewards for an epoch come in and a user already has a previous leaf for that `(epoch, account, token)` the Merkle tree updates that leaf with the cumulative amount.
4. The Merkle tree's leaves are unique by `(epoch, account, token)`, i.e., there can't be several leaves (differing in amounts only) for the same account, token, and epoch.

Recommendation: Consider documenting these properties in a "Merkle tree assumptions" section of the README. The 4th property can be verified in the `generateMerkleTree.js` script used to generate the Merkle Tree from a rewards JSON.

Furthermore, consider rephrasing this sentence of the README as it might be interpreted as unclaimed rewards being rolled over into the next epoch. Claims are only tracked cumulatively within the same epoch.

> The Merkle root can be updated, with claims tracked cumulatively across epochs.

MakerDAO: All the behaviors were documented in PR 12.

Cantina Managed: Verified.