MakerDAO: Op Farms

Security Review

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

September 9, 2024

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 Aug 29th to Aug 30th the Cantina team conducted a review of op-farms on commit hash f71a30e5.

The Cantina team reviewed MakerDAO's op-farms changes holistically on commit hash 119313bc and determined that all issues were resolved and no new issues were identified.

The team identified a total of **3** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 1
- Gas Optimizations: 0
- Informational: 2

3 Findings

3.1 Low Risk

3.1.1 L2 Farm's reward rate might not be constant

Severity: Low Risk

Context: Global Scope.

Description: An L1 DssVest contract defines a constant rate vesting schedule that an L1 VestedRewardsDistribution contract distributes to an L1FarmProxy contract whenever its minimum rewardThreshold is reached. The reward is bridged to the L2's L2FarmProxy contract. Once the L2FarmProxy's own rewardThreshold is reached, it notifies the L2 StakingReward contract which distributes the L2FarmProxy's reward balance over its StakingRewards.rewardDuration (rolling over any possible undistributed reward from the previous reward duration). Currently, L1FarmProxy and L2FarmProxy reward thresholds are set to the same value.

To achieve fair rewards for all stakers, a StakingReward's reward rate should be constant throughout its assigned DssVest vesting schedule. With the cross-chain setup there are several ways that can lead to non-constant reward rates.

Recommendation: Take the following into consideration:

- The L2 StakingReward's reward rate (essentially L2Proxy.rewardThreshold / StakingRewards.rewardsDuration) should be close to the DssVest's minting rate. The rewardThreshold and rewardsDuration configurations should be chosen to satisfy this.
- Keepers should monitor and call VestedRewardsDistribution.distribute on L1 and L2FarmProxy.forwardReward whenever their reward thresholds are reached.
- L2 sequencer downtimes or other bridging delays can lead to a delayed L2 distribution.
- Failed L2ProxyFarm reward token bridging transactions should be monitored and retried.

Maker: Acknowledged.

Cantina Managed: Acknowledged.

3.2 Informational

3.2.1 Additional verification checks for bridge farm scripts

Severity: Informational

Context: See below.

Description: The bridge initialization scripts check several assumptions on the involved contracts.

Recommendation: Consider adding additional checks:

- FarmProxyInit.sol#L99: cfg.vestBgn should be in the future, otherwise, the distribution will not be smooth and initial tokens will be lost when distributed to a farm without stakers.

Maker: Acknowledged. We decided not to add this verification. There could be cases where rewards are distributed immediately with the expectation that stakers will appear very fast. Also, the protocol has the ability to recover unused rewards from the farm.

Cantina Managed: Acknowledged.

3.2.2 Unnecessary check within the `FarmProxyInit`

Severity: Informational

Context: FarmProxyInit.sol#L92

Description: Within the `FarmProxyInit` the check against the `rewardThreshold` to be lower or equal than `type(uint224).max` is unnecessary because the `rewardThreshold` is already `uint224`.

Recommendation: Consider removing the unnecessary check.

Maker: Fixed in commit 119313bc.

Cantina Managed: Fixed.