Maker DAO - vote-delegate

Security Review

Cantina Managed review by:

Christoph Michel, Lead Security Researcher

M4rio.eth, Security Researcher

Shung, Associate Security Researcher

July 3, 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 May 31st to Jun 4th the Cantina team conducted a review of vote-delegate on commit hash d6e563f4.

The specific scope of the review included all files within the `src` directory.

The Cantina team reviewed MakerDao's vote-delegate changes holistically on commit hash 63700ad8724f339db0a595a612906a1c403b2069 and determined that all issues were resolved and no new issues were identified.

The team identified a total of **4** issues in the following risk categories:

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

3 Findings

3.1 Low Risk

3.1.1 Around 20% of locks can be blocked statistically

Severity: Low Risk

Context: VoteDelegate.sol#L86-L87

Description: When calling `reserveHatch`, `lock` calls happening within the next `HATCH_SIZE` blocks will revert. (This functionality exists to mitigate a liquidation-blocking issue.) Assuming `lock`s are randomly distributed over time, spamming `reserveHatch` as soon as the cooldown passed allows blocking `HATCH_SIZE / (HATCH_COOLDOWN + HATCH_SIZE + 1) = 19.23%` of all locks. An opposing vote delegate might be incentivized to block a certain percentage of locks to a vote delegate with opposing views.

Recommendation: As the user can retry their `lock` during the cooldown period and as the delegation is permanent for the new vote delegate contracts, we consider this attack only to have a low likelihood and impact. A reasonable `HATCH_SIZE` should be chosen that gives liquidators enough time to perform the liquidation, as well as a reasonable `HATCH_COOLDOWN` such that the lock-blocking ratio remains low while still allowing frequent liquidations. Currently, with 12s block times, the liquidator has 1 minute to liquidate after calling `reserveHatch`, but an unhealthy urn's liquidations can be blocked for 4 minutes. During this time the price could further deviate and the protocol risks ending up with bad debt.

Maker: This is the reason why the cooldown exists and why its size is relatively large vs the hatch size. Without changing the Chief there is no elegant solution for not blocking either locks or frees.

Cantina Managed: Acknowledged.

3.2 Gas Optimization

3.2.1 Hashed salt is unnecessary when creating a new instance of `VoteDelegate`

Severity: Gas Optimization

Context: VoteDelegateFactory.sol#L62

Description: In `VoteDelegateFactory.create()` function, `create2` method is used to create a new instance of `VoteDelegate`. The `create2` uses the `keccak256` hash of `msg.sender` as the salt value. This hashing operation is redundant because the `msg.sender` can be directly used as the salt.

Recommendation: Consider replacing `salt: keccak256(abi.encode(msg.sender))` with `salt: bytes32(uint256(uint160(msg.sender)))` and updating `getAddress()` accordingly.

Maker: Changed in commit 63700ad8.

Cantina Managed: Fixed.

3.3 Informational

3.3.1 `reserveHatch` cooldown might be off-by-one

Severity: Informational

Context: VoteDelegate.sol#L86-L87

Description: With the current `HATCH_SIZE`, `HATCH_COOLDOWN`, and `block.number > hatchTrigger + HATCH_SIZE + HATCH_COOLDOWN` check in `reserveHatch`, the locking and cooldown periods look as follows:

- Block $n + 0$: call `reserveHatch`: **can** call lock.
- Block $n + (1 - 5)$: **cannot** lock.
- Block $n + (6 - 25)$: **can** lock.
- Block $n + 26$: can call `reserveHatch` again, repeats.

When spamming `reserveHatch` as soon as the cooldown is over, users cannot lock 5 (1-5) out of 26 blocks (0-25). This is a ratio of `HATCH_SIZE / (HATCH_SIZE + HATCH_COOLDOWN + 1) = 5/26 = 19.23%`. One can

call `reserveHatch` again after `HATCH_SIZE + HATCH_COOLDOWN + 1` blocks. It might have been intended to be 20% and `HATCH_SIZE + HATCH_COOLDOWN`.

Recommendation: Consider using >= instead of > in the `reserveHatch` cooldown check:

```
  function reserveHatch() external {
-     require(block.number > hatchTrigger + HATCH_SIZE + HATCH_COOLDOWN, "VoteDelegate/cooldown-not-finished");
+     require(block.number >= hatchTrigger + HATCH_SIZE + HATCH_COOLDOWN, "VoteDelegate/cooldown-not-finished"
);
      hatchTrigger = block.number;

      emit ReserveHatch();
  }
```

Maker: Changed in commit 63700ad8.

Cantina Managed: Fixed.

3.3.2 IOUs locked in vote delegate instead of sent to user

Severity: Informational

Context: VoteDelegate.sol#L89, Old VoteDelegate.sol#L86

Description: Unlike the existing vote delegate contracts, the new contract does not transfer the IOU tokens received from the `chief` to the delegating user. It was stated that the IOUs are obsolete and not planned to be used.

Recommendation: Consider documenting this difference in IOU behavior, either in the Lockstake Engine vote delegate changes section 3a) or in the vote delegate repository.

Maker: Fixed, a comment has been added in the readme (see commit 5132023e):

```
In order to simplify the logic, the IOU tokens generated by DSChief are kept in the new VoteDelegate contract.
```

Cantina Managed: Verified.