MakerDAO: Endgame Toolkit Updates

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.

On Sep 2nd the Cantina team conducted a review of endgame-toolkit on commit hash eb49fa61.

The Cantina team reviewed MakerDAO's endgame-toolkit changes holistically on commit hash 14268515 and determined that all issues were resolved and no new issues were identified.

The team identified a total of **6** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 1
- Gas Optimizations: 0
- Informational: 5

3 Findings

3.1 Low Risk

3.1.1 DoS on init spell by staking contract

Severity: Low Risk

Context: UsdsSkyFarmingInit.sol#L52-L55, Usds01PreFarmingInit.sol#L36-L39

Description: The farming init scripts check that the farm (`StakingRewards` contract)'s `lastUpdateTime` is 0:

```
require(
    StakingRewardsLike(p.rewards).lastUpdateTime() == 0,
    "UsdsSkyFarmingInit/rewards-last-update-time-invalid"
);

require(
    StakingRewardsLike(p.rewards).lastUpdateTime() == 0,
    "Usds01PreFarmingInit/rewards-last-update-time-invalid"
);
```

The deployment of these contracts happens in a previous step, and anyone can call `p.rewards.getReward()` which calls `updateReward(msg.sender)`, setting `lastUpdateTime` to the current time. (Users could also already `stake` if the staking token is available at deployment.)

The check in the spell will revert and the initialization will fail.

Recommendation: The intent of this `require` call is to ensure that the rewards farm is freshly deployed and hasn't received previous rewards. Consider checking `rewards.rewardsDistribution() == 0` and `rewards.rewardRate() == 0` instead as these state variables can only be set by admin fields, not by users.

Maker: Resolved in commit 14268515.

Cantina Managed: Fixed.

3.2 Informational

3.2.1 Incorrect revert message in the `03-UsdsSkyFarmingCheck.s.sol`

Severity: Informational

Context: 03-UsdsSkyFarmingCheck.s.sol#L44

Description: The revert message that checks if the staking token is usds is incorrect as it should state that the staking-token is invalid.

```
require(StakingRewardsLike(rewards).stakingToken() == usds, "StakingRewards/invalid-rewards-token");
```

Recommendation: Correct the revert message `StakingRewards/invalid-staking-token`.

Maker: Fixed in commit 14268515.

Cantina Managed: Verified.

3.2.2 Unnecessary tokens check in the `UsdsSkyFarmingInit`

Severity: Informational

Context: UsdsSkyFarmingInit.sol#L46

Description: The following check is unnecessary because it is already existent in the constructor of the Staking Rewards.

```
require(stakingToken != rewardsToken, "UsdsSkyFarmingInit/rewards-token-same-as-staking-token");
```

Recommendation: Consider removing this check.

Maker: Fixed in commit 14268515.

Cantina Managed: Fixed.

3.2.3 Additional verification checks for farm scripts

Severity: Informational

Context: See below.

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

Furthermore, a `Phase1b_UsdsSkyFarmingCheckScript` script was created that should do a sanity check on a deployed instance. Currently the sanity checks are incomplete, the missing pieces are: `Vest.cap`, `Vest.tau`, `Vest.bgn` and `Vest.tot`.

Recommendation: Consider adding additional checks:

- UsdsSkyFarmingInit.sol#L74: `p.vestBgn` should be the start date minus the rewards duration, otherwise, the initial rewards rate differs from the mint rate. Then, users should already be able to stake prior to the start date, otherwise, initial tokens will be lost when distributed to a farm without stakers.
- Add the following checks within the `Phase1b_UsdsSkyFarmingCheckScript`:

```
require(DssVestWithGemLike(vest).cap() == {vestCap}, "DssVest/invalid-cap");
require(DssVestWithGemLike(vest).bgn(vestId) == vestBgn, "DssVest/invalid-bgn");
require(DssVestWithGemLike(vest).tot(vestId) == vestTot, "DssVest/invalid-tot");
require(DssVestWithGemLike(vest).fin(vestId) == vestBgn + vestTau, "DssVest/invalid-tau");
```

Maker: We decided that the ownership of the `DssVest` contract lies outside this repo, so any contract-level parameter checks are out of the scope.

Cantina Managed: The `Phase1b_UsdsSkyFarmingCheckScript` script now check `vest.bgn`, `vest.fin` and `vest.tot` against pre-configured values. Further checks on the `cap` and reward rates will be performed in the main spell.

3.2.4 Scripts don't set `rewardsDuration` on farm

Severity: Informational

Context: UsdsSkyFarmingInit.sol#L69, Usds01PreFarmingInit.sol#L44

Description: The scripts and the corresponding spell never set a reward duration on the farm contract (`StakingRewards` instance). It currently defaults to a 7-day reward duration.

Recommendation: Confirm the 7-day reward duration or consider explicitly setting it.

Maker: Acknowledged. We confirm the 7-day reward duration.

Cantina Managed: Acknowledged.

3.2.5 Scripts don't grant SKY minting rights to vesting contract

Severity: Informational

Context: UsdsSkyFarmingInit.sol#L39

Description: The `vest` contract (instance of `DssVestMintable` used for farming SKY reward tokens needs minting rights for the SKY token. This is currently not done in the presented scripts.

Recommendation: Consider adding minting rights to the `vest` during the initialization phase. This has to be done through a spell. Also consider setting an appropriate `cap` for minting (before creating the vesting schedule), currently, the cap is set to `type(uint256).max`.

Maker: Acknowledged. This is done in the top-level spell being worked on. This is a 1-of setup, while farm scripts are meant to be executable more than once.

Cantina Managed: Acknowledged. Maker intends to also set an appropriate vest cap in the top-level spell.