MakerDAO: SP-Beam

Security Review

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

April 1, 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 sp-beam on commit hash 1d081615. The team identified a total of **10** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 1
- Medium Risk: 0
- Low Risk: 2
- Gas Optimizations: 0
- Informational: 7

3 Findings

3.1 High Risk

3.1.1 `DSPC.set` can bypass the `cfg.step` constraint by including the same `id` multiple times in the `updates` array and continuously increase the `bps`

Severity: High Risk

Context: DSPC.sol#L268

Description: The `DSPC.set` function allows a `bud` (whitelisted actor) to modify the rates for `SSR`, `DSR`, or any collateral stability fee. A `ward` can restrict this action by defining `min`, `max`, and `step` parameters for the individual rates.

The `step` parameter specifies the maximum value by which a rate can be increased in a single call.

```
// Calculates absolute difference between the old and the new rate
uint256 delta = bps > oldBps ? bps - oldBps : oldBps - bps;
require(delta <= cfg.step, "DSPC/delta-above-step");
```

The next `set` call should be only possible after a freeze period defined by `tau`. However, a `bud` can bypass the `cfg.step` constraint by including the same `id` multiple times in the `updates` array and continuously increase the `bps`.

Each loop iteration updates the rate, and the subsequent iteration incorrectly interprets it as the old rate. This would allow to set the `rate` to the maximum in one single call.

Recommendation: The `set` function could enforce the `updates` array to be sorted, which would easily detect duplicates in the `updates` array.

Maker: Fixed in commit 43ed724.

Cantina Managed: Fixed.

3.2 Low Risk

3.2.1 DSPC could lead to undesirable configuration of rates

Severity: Low Risk

Context: DSPC.sol#L252-L264

Description: It is possible for Governance to set a rate directly in DSS that is outside the configured `(min,max)` for the DSPC. Further, it is possible for a `bud` to frontrun a `(min,max)` change spell by calling `set` in such a way that after the spell is cast the rate would be outside the new `(min,max)`. These scenarios result in an undesirable configuration where either the DSPC is locked because the rate is more than `step` away from the new `(min,max)` range or just where the rate is outside the range because `bud` does not call `set`.

Recommendation: Ensure that when a DSPC is active, governance does not directly change the `(min,max)` config unless it also checks/resets the rate in DSS to be within that range. Further protections could be added to DSPC to handle this edge case either by using the `(min,max)` for the `oldBPS` when `oldBPS` is out of range or in that edge case using the appropriate `min` or `max` as the new rate.

- Option 1:

```
// Check rate change is within allowed gap
uint256 oldBps;
if (id == "DSR") {
    oldBps = conv.rtob(PotLike(pot).dsr());
} else if (id == "SSR") {
    oldBps = conv.rtob(SUSDSLike(susds).ssr());
} else {
    (uint256 duty,) = JugLike(jug).ilks(id);
    oldBps = conv.rtob(duty);
}

if (oldBps < cfg.min) {
    oldBps = cfg.min;
} else if (oldBps > cfg.max) {
    oldBps = cfg.max;
}

// Calculates absolute difference between the old and the new rate
uint256 delta = bps > oldBps ? bps - oldBps : oldBps - bps;
```

  This means that the rate after `set` could be within `(min, min + step)` if the rate was under `min` or `(max - step, max)` if it was over.

- Option 2:

```
if (oldBps < cfg.min) {
    bps = cfg.min;
} else if (oldBps > cfg.max) {
    bps = cfg.max;
} else {
    // Calculates absolute difference between the old and the new rate
    uint256 delta = bps > oldBps ? bps - oldBps : oldBps - bps;
    require(delta <= cfg.step, "DSPC/delta-above-step");
}
```

  This means that the rate after `set` could be `min` if the rate was under `min` or `max` if it was over.

Maker: It would be undesirable for a governance to indirectly disable this module. It would be better for this module to behave like the Autoline module: If governance sets a value outside `(min, max)` without disabling this module, this module will be able to override the rate to be back within the parameters of this module.

The Governance spell process is mature and monitored by multiple teams so this edge case should be rare.

Fixed in commit c9dfdfb5.

Cantina Managed: This change addresses the concern of manually set parameters outside DSPC cfg.

3.2.2 `rate` and `rate constraints` can be set for non-existing `ilk` in `jug`

Severity: Low Risk

Context: DSPC.sol#L280

Description: In the current design, the `DSPC.set` function allows setting a `stability fee` for a non-existing `ilk` collateral in the `jug` contract. There is no validation check to ensure that the `ilk` collateral exists in the `jug` contract.

This is also true for the rate constraints (cfg) in the `file` method. The contract can have constraints for non-existing collateral type.

Recommendation: It should only be possible to set a stability fee for an existing collateral type. The `file` function could validate whether a collateral type exists before allowing the definition of rate constraints. Without specifically defined rate constraints (`cfg`) the set function would revert.

```
(uint256 duty,) = jug.ilks(id);
 require(duty > 0, "DSPC/ilk-not-initialized");
```

Maker: Fixed in commit c9dfdfb.

Cantina Managed: Fixed.

3.3 Informational

3.3.1 Multiple `bud`s could cause frontrun issue

Severity: Informational

Context: DSPC.sol#L241-L243

Description: Because of the `tau + toc` check in `set`, multiple `bud`s could intentionally or unintentionally frontrun calls to `set`. If `bud-a` calls to lower the rate and just before that is executed, `bud-b` raises it, `bud-b`'s call would reset the `toc` and cause `bud-a`'s transaction to revert. Worst case, this could open the system to a highly motivated `bud` front-running all other transactions and essentially forcing the rate to stay close to `min`/`max`.

Recommendation: Consider the trust assumptions and monitoring of `bud`s and `diss` any `bud` that appears to be misbehaving.

Maker: Acknowledged, only one bud will be in place. Even if we ever chose to go with more than one, buds are highly trusted entities.

Cantina Managed: Acknowledged.

3.3.2 Validating MOM file could be more explicit/clear

Severity: Informational

Context: DSPC.sol#L195-L197

Description: While the current code is slightly more gas efficient, this admin function is not a commonly traversed function (it will only be used by an emergency spell through the MOM or by governance's normal process to turn on/off the DSPC), the current code is not as clear/self-documenting as possible. `bad` is meant to be either `0` or `1`, but the check allows any value between 0 and 1.

Recommendation: It would be more clear/explicit about the intent of `bad` to use something like the following check:

```
require(data == 1 || data == 0, "DSPC/invalid-bad-balue");
```

Maker: Fixed in commit a67531bb.

Cantina Managed: Fixed confirmed.

3.3.3 DSPC Deployment could finalize authorization rather than waiting for Init

Severity: Informational

Context: DSPCDeploy.sol#L58-L63

Description: The `DSPCDeploy` script could finalize the authorization for the new contracts (`rely`ing the DSCP on the MOM and giving Governance authority over the MOM). These actions are currently handled by the init library, but it would be possible to move them to before the owners are switched on deploy. This would result in the deployment ending with a more complete system including required authorizations and the init spell would be able to focus on authorizing the system into DSS and configuring the ilks.

Maker: Acknowledged, the default approach is to just give the `PauseProxy` ownership of the deployed contracts and handle everything else in the initialization, as it's supposed to happen within a spell.

This decreases the possibility of initialization bugs as the spell process is very mature and under the scrutiny of multiple teams.

Cantina Managed: Acknowledged.

3.3.4 DSPC module defines DSR and SSR id in upper case while pot and susds are using lower case

Severity: Informational

Context: DSPC.sol#L272

Description: The DSPC module is used to identify the DSR rate with the ID DSR and the SUSDS rate with the ID SSR. However, the corresponding contracts use lowercase identifiers for these IDs.

```
if (id == "DSR") {
    pot.drip();
    pot.file("dsr", ray);
} else if (id == "SSR") {
    susds.drip();
    susds.file("ssr", ray);
```

Recommendation: Consider using lowercase for the IDs to maintain consistency with the contract identifiers. This would simplify contract calls.

```
if (id == "dsr") {
    pot.drip();
    pot.file("id, ray);
} else if (id == "ssr") {
    susds.drip();
    susds.file(id, ray);
```

Maker: Acknowledged. id follows the collateral naming convention for ilks (i.e: "ETH-A") so we chose to keep the same convention for DSR and SSR. On file we're also following the convention used in other contracts in the protocol (all lower case for the param being changed).

Cantina Managed: Acknowledged.

3.3.5 No specific revert message if _cfgs is not set for a rate id

Severity: Informational

Context: DSPC.sol#L253

Description: The cfg stores the rate constraint for a specific rate id. These rate constraints are checked in the set function. If no cfg is set for a specific id, the current implementation will always revert with require(bps <= cfg.max, "DSPC/above-max");.

Recommendation: Consider using a specific revert message if the cfg constraints are not set for a specific rate id. Currently, if minimum and maximum are set but no step is defined, it will revert with DSPC/delta-above-step.

Maker: Fixed in commit c9dfdfb.

Cantina Managed: Fixed.

3.3.6 An ilk collateral can't have the names DSR or SSR for the DSPC module to work correctly

Severity: Informational

Context: DSPC.sol#L257

Description: The DSPC function set doesn't distinguish between saving rates and stability fees for collateral. The input is a updates array containing an id. Based on the id the correct contract will be called.

```
if (id == "DSR") {
    oldBps = conv.rtob(PotLike(pot).dsr());
} else if (id == "SSR") {
    oldBps = conv.rtob(SUSDSLike(susds).ssr());
} else {
    (uint256 duty,) = JugLike(jug).ilks(id);
    oldBps = conv.rtob(duty);
}
```

However, this pattern assumes ilk collateral will never be named DSR or SSR. While this is unlikely, it is worth mentioning as a potential edge case.

Recommendation: This restriction should be documented.

Maker: Acknowledged. We're aware of this issue, but historically all ilks have a -{A,B,C,...} suffix, so this collision seems impossible in practice.

Cantina Managed: Acknowledged.

3.3.7 `toc` could be initialized in the `constructor` with `block.timestamp` to apply the `tau` delay for the first set call

Severity: Informational

Context: DSPC.sol#L243

Description: The `tau` variable defines a time delay between two `set` calls. In each `set` call, a second variable called `toc` is updated with `block.timestamp` to enforce the `block.timestamp >= tau + toc` delay for the next `set` call.

```
require(block.timestamp >= tau + toc, "DSPC/too-early");
toc = uint128(block.timestamp);
```

However, this `delay` won't apply to the first call `set` call. After the contracts are deployed and whitelisted by Maker/Sky governance with a `spell`, the rates could be changed immediately.

Recommendation: If the first `set` call should have a delay as well. Initialize `toc` with the current `block.timestamp` in the constructor.

Maker: Acknowledged. Due to the permissioned nature of the module, this could be easily solved at the social layer. The onboarding process includes a governance vote and GSM delay, making immediate rate changes post-spell expected and manageable.

Cantina Managed: Acknowledged.