Sky: SBE Beam

Security Review

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

July 14, 2026

Contents

1 Introduction **2**
1.1 About Cantina . . . 2
1.2 Disclaimer . . . 2
1.3 Risk assessment . . . 2
1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**

3 Findings **4**
3.1 Informational . . . 4
3.1.1 Instead of using a cached `farmOwner` in `SBEBeam`, consider fetching the real owner at run time . . . 4
3.1.2 Minor issues, Typos & Documentation . . . 4

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 level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

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 are 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

Sky Protocol is a decentralised protocol developed around the USDS stablecoin.

From July 11th to July 13th the Cantina team conducted a review of dss-flappers on commit hash 9f479253. The team identified a total of **2** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	0	0	0
Gas Optimizations	0	0	0
Informational	2	2	0
Total	**2**	**2**	**0**

The Cantina Managed team reviewed Sky's dss-flappers holistically on commit hash 655d2dd2 and concluded that all findings were addressed and no new vulnerabilities were identified.

3 Findings

3.1 Informational

3.1.1 Instead of using a cached `farmOwner` in `SBEBeam`, consider fetching the real owner at run time

Severity: Informational.

Context: SBEBeam.sol#L120.

Description: `SBEBeam` caches a local `farmOwner` via `file` which can be stale or mismatch the current `splitter`'s `farm`'s owner for other reasons.

Note that whenever `farmOwner` is used in `set`, it already assumes the configured `splitter.farm()` matches and the owner is `FarmOwner`-compatible; otherwise the rewards duration update call will revert. Therefore, one could consider just fetching the current `farmOwner` during `set`.

The change in behavior is that currently a stale `farmOwner` acts as an additional DoS on the `SBEBeam` but this might be unnecessary or even considered misbehavior: If the splitter later `file`s a new farm, `SBEBeam` should arguably follow that change because its configurable throughput parameters are `splitter` / `kicker`-bound, while the concrete farm is an implementation detail.

Recommendation: Consider not caching `farmOwner`. Instead, in `set`, when rewards are enabled (`burn<WAD`), fetch `splitter.farm()` and its current `owner()`, sanity-check that the owner is `FarmOwner`-like for that farm, and use it to update rewards duration if needed.

For example, similar to the following pseudo code:

```
// in set:
// if rewards are enabled, a farm is required
if (burn < WAD) {
  address farm = splitter.farm();

  if (hop != farm.rewardsDuration()) {
    address farmOwner = farm.owner();
    // owner is assumed to be FarmOwner-like in this case
    // optional sanity check
    require(farmOwner.farm() == farm, "SBEBeam/farm-sanity-failed");
    farmOwner.setRewardsDuration(hop);
  }
}
```

Sky: Fixed in commit 655d2dd2.

Cantina Managed: Fix verified.

3.1.2 Minor issues, Typos & Documentation

Severity: Informational.

Context: (*See each case below*).

Description:

1. SBEBeam.sol#L183: The farm is fetched through `farmOwner`, which may not always be defined, while `splitter` is always defined and exposes `splitter.farm()`. This is safe in the current code path because the farm is only looked up when `farmOwner` is defined and the preceding `require` ensures the farms match, but it makes the lookup less direct and slightly harder to reason about. Consider using `splitter.farm()` instead.
2. SBEBeam.sol#L170: Consider also checking `splitter.farm().owner()==farmOwner` as, currently, an old, nominated-but-not-accepted, or misconfigured `farmOwner` can pass all checks, then later revert in `setRewardsDuration`'s auth. This produces a less helpful error message.

3. SBEBeam.sol#L48: The term ”burn rate” can be misleading because it suggests only the burned `lot` amount is capped per `hop`, while the value also affects the reward component (entire `kbump` as the comment says). Use clearer terminology in the comment such as ”kick rate” or ”surplus throughput”.
4. FlapperInit.sol#L315: The initialization checks can be strengthened by verifying that `farmOwner.farm()` matches the expected `farm` address. This helps ensure the passed farm owner is associated with the intended live farm.
5. FarmOwner.sol#L94 - In the farm implementation the `safeTransfer` is used and in the `FarmOwner` the `transfer` is used which creates an inconsistency, we can assume that if safe transfer works on farm ⇒ farmOwner would work for farmOwner ⇒ to but in case that is not the case, the tokens will be lost in the `farmOwner`.
6. SBEBeam.sol#L47 - The `minHop` can be uint64 and moved near `tau` to get it packed in one slot.

Recommendation: Consider addressing the issues mentioned above.

Sky: 1, 2 and 3 should be covered in commit 655d2dd2. For the rest, we are leaving as it is.

Cantina Managed:

1. Fixed: not relevant anymore because of the fix for the caching issue.
2. Fixed: not relevant anymore as `farmOwner` is by definition `farm.owner()` now.
3. Fixed.