CANTINA

Sky: Spark ALM Controller

Security Review

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

August 18, 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

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

From Jul 28th to Aug 8th the Cantina team conducted a review of spark-alm-controller on commit hash c0292396. The team identified a total of **3** issues in the following risk categories:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	1	1	0
Gas Optimizations	0	0	0
Informational	2	1	1
Total	**3**	**2**	**1**

The Cantina Managed team reviewed Sky's `spark-alm-controller` holistically on commit hash 83defcbd and concluded that all the issues were addressed and no new vulnerabilities were identified.

3 Findings

3.1 Low Risk

3.1.1 Withdrawing from ERC4626 and Aave requires deposit rate limits to be set

Severity: Low Risk

Context: MainnetController.sol#L319

Description: When withdrawing by calling `withdrawERC4626`/`redeemERC4626`/`withdrawAave`, the `WITHDRAW` limit is consumed and the corresponding `DEPOSIT` limits are reset:

```
// withdrawERC4626
_rateLimitedAsset(LIMIT_4626_WITHDRAW, token, amount);
_cancelRateLimit(RateLimitHelpers.makeAssetKey(LIMIT_4626_DEPOSIT, token), amount);
// rateLimits.triggerRateLimitIncrease(key, amount);
```

However, this requires that the corresponding `DEPOSIT` limit is non-zero as the rate limiter checks for it:

```
function triggerRateLimitIncrease(bytes32 key, uint256 amountToIncrease)
    external
    override
    onlyRole(CONTROLLER)
    returns (uint256 newLimit)
{
    RateLimitData storage d = _data[key];
    uint256 maxAmount = d.maxAmount;

    require(maxAmount > 0, "RateLimits/zero-maxAmount");
    // ...
}
```

This can become a problem when off-boarding assets, when one might want to disable deposits and only enable withdrawals.

Recommendation: Be aware of this restriction and that a tiny non-zero deposit limit might have to be set if withdrawals should continue to work.

Sky: Fixed in PR 138.

Cantina Managed: Fix verified.

3.2 Informational

3.2.1 Improved documentation

Severity: Informational

Context: README.md

Description: Currently we don't have one place in the docs that shows which rate-limit each function uses and how it changes it. This makes reviews and audits hard. Some functions decrease a limit, others increase it, and some only check that a limit exists. We also have similar flows that don't handle limits the same way, e.g. of a limits matrix

Function	Limits Decremented	Limits Incremented
`depositPSM`	`LIMIT_PSM_DEPOSIT`	---
`withdrawPSM`	`LIMIT_PSM_WITHDRAW`	---
`depositERC4626`	`LIMIT_4626_DEPOSIT`	---
`withdrawERC4626`	`LIMIT_4626_WITHDRAW`	`LIMIT_4626_DEPOSIT`
`redeemERC4626`	`LIMIT_4626_WITHDRAW`	`LIMIT_4626_DEPOSIT`
`depositAave`	`LIMIT_AAVE_DEPOSIT`	---
`withdrawAave`	`LIMIT_AAVE_WITHDRAW`	`LIMIT_AAVE_DEPOSIT`

Recommendation: Consider adding a limits matrix for every controller.

Sky: Added a programmatic verification script to check all the rate limits using AST in PR 137.

Cantina Managed: Verified.

3.2.2 Cannot perform a pure staking farm rewards claim

Severity: Informational

Context: MainnetController.sol#L878

Description: To receive the rewards from the staking farm, the `withdrawFromFarm(farm, usdsAmount)` function must be called. It performs both a withdrawal and a claim of the rewards (`getReward`):

```
proxy.doCall(
    farm,
    abi.encodeCall(IFarmLike.withdraw, (usdsAmount))
);
proxy.doCall(
    farm,
    abi.encodeCall(IFarmLike.getReward, ())
);
```

Note that it can't be claim-only with a zero-amount withdrawal as the `farm.withdraw` function reverts for zero amounts:

```
function withdraw(uint256 amount) public nonReentrant updateReward(msg.sender) {
    require(amount > 0, "Cannot withdraw 0");
    _totalSupply = _totalSupply - amount;
    _balances[msg.sender] = _balances[msg.sender] - amount;
    stakingToken.safeTransfer(msg.sender, amount);
    emit Withdrawn(msg.sender, amount);
}
```

Recommendation: If claim-only functionality is desired, consider guarding the `IFarmLike.withdraw` with a zero-amount check:

```
if (usdsAmount > 0) {
    proxy.doCall(
        farm,
        abi.encodeCall(IFarmLike.withdraw, (usdsAmount))
    );
}
```

Note that this requires that `LIMIT_FARM_WITHDRAW` rate limits remain non-zero. If this is expected to be an issue, a non-rate-limited function could be added that only performs the reward claim.

Sky: We actually noticed this too during our development process and added a specific claim function but went over size. This was the only way we could fit it in without reatrchitecting.

Cantina Managed: Acknowledged.