Sky: Spark ALM Controller v1.8

Security Review

Cantina Managed review by:

Christoph Michel, Lead Security Researcher
Mario.eth, Lead Security Researcher

November 13, 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 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 Oct 20th to Oct 22nd the Cantina team conducted a review of spark-alm-controller on commit hash 3144c913 (tag `v1.8.0-beta.0`). The team identified a total of **8** issues:

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	7	4	3
Total	**8**	**5**	**3**

The Cantina Managed team reviewed Sky's `spark-alm-controller` holistically on commit hash 7be95937 (tag `v1.8.0`) and concluded that all findings were addressed and no new vulnerabilities were identified.

2.1 Low Risk

2.1.1 ETH received from exchange in `OTCBuffer` will be lost

Severity: Low Risk

Context: OTCBuffer.sol#L34

Description: The `OTCBuffer` contract can receive ETH from the exchange as the result of a swap as it implements a `receive() payable` but the received ETH will be locked and lost in the contract as there's no functionality to send it to the ALM Proxy.

Recommendation: The `MainnetController` OTC swap claim functionality currently only supports USD-equivalent stablecoins. ETH should never be the swap output token. Remove the `receive() payable` function such that the transfers from the exchange fail instead of the ETH being locked & lost in the contract.

Sky: Fixed in PR 176.

Cantina Managed: Fixed.

2.2 Informational

2.2.1 `WSTETH_WITHDRAW_QUEUE` withdrawals limits affect the `requestWithdrawFromWstETH`

Severity: Informational

Context: MainnetController.sol#L370-L400

Description: The function that requests `wstETH` withdrawals from `WSTETH_WITHDRAW_QUEUE` submits a single request amount, that amount is bounded by Lido withdrawal queue limits. The withdrawal queue enforces a minimum and maximum per request:

```
uint256 public constant MIN_STETH_WITHDRAWAL_AMOUNT = 100;

/// @notice maximum amount of stETH that is possible to withdraw by a single request
/// Prevents accumulating too much funds per single request fulfillment in the future.
/// @dev To withdraw larger amounts, it's recommended to split it to several requests
uint256 public constant MAX_STETH_WITHDRAWAL_AMOUNT = 1000 * 1e18;
```

The current implementation always sends an array with one element which means that large withdrawals cannot exceed the max per request, so they must be split into multiple requests.

Recommendation: Split large withdrawals into multiple requests and ensure each per request amount is within the allowed bounds. Either loop in the relayer to submit multiple requests or mirrors the approach used for CCTP where transfers are split by a limit and the looping is performed in the contract itself.

Sky: Acknowledged. We're not going to make a code change here, we will just make multiple calls from the relayer if needed.

Cantina Managed: Acknowledged.

2.2.2 Hardcode the `spender` to ALM Proxy in `OTCBuffer`

Severity: Informational

Context: OTCBuffer.sol#L24

Description: The `OTCBuffer` receives the swap result from an OTC exchange. The ALM Proxy claims it via `MainnetController.claimOTC` which requires an approval from the `OTCBuffer`. This approval can be set by the `DEFAULT_ADMIN` (expected to be `SPARK_PROXY`) via the `OTCBuffer.approve` method:

```
function approve(address asset, address spender, uint256 allowance)
    external onlyRole(DEFAULT_ADMIN_ROLE)
{
    IERC20(asset).forceApprove(spender, allowance);
}
```

Recommendation:

1. As the `spender` should always be the ALM Proxy, consider hardcoding the `spender` to the ALM Proxy instead of receiving an arbitrary spender parameter. This can be done by receiving the `almProxy` address in the constructor and saving it to a `address public immutable almProxy`.
2. Optionally, also hardcode the amount to `uint256.max`. The trade-off is that the `DEFAULT_ADMIN` cannot revoke an existing approval (and thus claiming the swap result) to the `almProxy`, hardening security against compromised `DEFAULT_ADMIN`s, however, if there are issues with the `MainnetController`/ALMProxy implementation, revoking approvals by an honest `DEFAULT_ADMIN` might be beneficial.

The changes reduce trust in a third-party `DEFAULT_ADMIN` and make it easier to verify the correctness of the `OTCBuffer` upon deployment.

Sky: Fixed in PR 179.

Cantina Managed: Fixed.

2.2.3 Quantifying the max loss for an OTC exchange

Severity: Informational

Context: MainnetController.sol#L957

Description: The OTC exchange happens asynchronously; the ALM Proxy sends ETH to a whitelisted exchange address and is supposed to receive the swap result in an `OTCBuffer` contract. The exchange needs to be trusted. To reduce risk, there are two rate limits in play:

1. The standard `RateLimits.sol` rate limit per `exchange`.
2. The `isOtcSwapReady(exchange)` net swap limit per `exchange`. This is the max amount of exposure for an exchange. This amount is netted and reset only when receiving the max result. There's also a recharge limit per second and a slippage percentage to ensure the `OTCSwap` functionality does not permanently become stuck if the exchange sends fewer tokens than expected.

> This will provide strong guarantees to Spark/Sky that at most `X` can be stolen/lost, per whitelisted OTC route, while still allowing for rapid throughput into an offchain market with high liquidity such as Binance.

The `X` referred to in this README is a combination of the two rate limits, and depends on the `OTC_LIMIT`, the slippage parameter, and the time elapsed (because of the recharge rate). Each block `blockTime * rechargeRate / maxSlippage` can be taken out without impacting the OTC rate limit (`isOtcSwapReady` will still return `true` the next block). In addition, the maximum that can be taken out with a single OTC is `LIMIT_OTC_SWAP`.

```
max loss per time elapsed: LIMIT_OTC_SWAP + (timeElapsed * rechargeRate) / maxSlippage
```

Recommendation: No action required. Consider quantifying the max loss per time period as above and documenting it.

Sky: Acknowledged. We think documentation is sufficient as is.

Cantina Managed: Acknowledged.

2.2.4 Missing OTC check in `setOTCWhitelistedAsset`

Severity: Informational

Context: MainnetController.sol#L278-L286

Description: The `setOTCWhitelistedAsset` allows configuring a whitelisted asset for an exchange even if the OTC entry for that exchange has not been initialized. This can leave whitelist entries set for non existent exchanges where no buffer has been configured. This currently does not poses a risk because `otcSend` will fail if the buffer is not set.

Recommendation: Add a sanity check that the OTC entry exists before allowing whitelist changes.

Sky: Fixed in PR 180.

Cantina Managed: Fixed.

2.2.5 The `ForeignControllerInit` missing `maxSlippage` set

Severity: Informational

Context: ForeignControllerInit.sol#L163-L165, MainnetControllerInit.sol#L183-L185

Description: The `MainnetControllerInit` wires max slippage at initialization and upgrade by accepting a MaxSlippageParams[] and iterating to call `setMaxSlippage` on the controller. ForeignControllerInit does not do the same.

Recommendation: Add max slippage configuration to `ForeignControllerInit` mirroring `MainnetControllerInit`.

Sky: Fixed in PR 178.

Cantina Managed: Fixed.

2.2.6 The `setOTCBuffer` and `setOTCRechargeRate` could be combined into one function

Severity: Informational

Context: MainnetController.sol#L255-L276

Description: Two admin functions `setOTCBuffer` and `setOTCRechargeRate` update fields in the same OTC struct. These can be combined into a single function to reduce the contract size. Note that the `rechargeRate18` value is expected to be updated frequently; combining the functions will require always specifying the buffer parameter when updating the rate, which adds overhead.

Recommendation: Combine the two functions into one function.

Sky: Acknowledged. We will keep in the same set up as the current implementation.

Cantina Managed: Acknowledged.

2.2.7 Documentation & minor issues

Severity: Informational

Context: *(See each case below)*

Description:

- MainnetController.sol#L182: The `mintRecipients` variable should note that it's being used for CCTP `// CCTP mint recipients`
- MainnetController.sol#L949: Missing note `// NOTE: This will lose precision for tokens with >18 decimals.`
- MainnetController.sol#L161: The `buffer` variable is now ambiguous because we have 2 buffer variables in the contract: `buffer` and `OTC.buffer`. Consider adding a note to this `buffer` variable that this is the allocator buffer
- MainnetController.sol#L1074: The error says "transfer-failed". Consider if it should be "transferFrom-failed" instead.

Recommendation: Consider fixing the issues mentioned above.

Sky: Fixed in PR 177.

Cantina Managed: Fixed.