CANTINA

Bitcorn oft

Security Review

Cantina Managed review by:

Eric Wang, Lead Security Researcher
RustyRabbit, Security Researcher

March 8, 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

Corn is a layer 2 network focused on revolutionizing the capital efficiency of Bitcoin as a nascent asset class. Designed to provide scalable infrastructure that leverages Ethereum in a manner that allows for the secure management of billions of dollars in liquidity with low transactional costs secured by the Bitcoin L1.

On Feb 5th the Cantina team conducted a review of bitcorn-oft on PR 35. The team identified a total of **3** 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	2	1	1
Total	**3**	**2**	**1**

3 Findings

3.1 Low Risk

3.1.1 Pausing the contract does not pause voting delegations

Severity: Low Risk

Context: CornOFT.sol#L113-L120

Description: The `CornOFT` contract overrides the `_update()` function to include a `whenNotPaused` modifier. As a result, a privileged role can pause the contract to prevent ERC-20 token transfers between accounts on the current chain or cross-chain, or prevent privileged roles from minting or burning tokens.

However, it should be noted that voting delegations are still allowed when the contract is paused. There could be a situation where a user with a large number of tokens gets compromised, and the attacker gets the user to delegate themselves, allowing them to vote for malicious proposals and making them easier to pass.

Recommendation: Consider clarifying whether this is the intended design. If so, consider adding code specifications to clarify which actions should be paused when the contract is paused. If not, consider overriding the `_moveDelegateVotes()` from the `VotesUpgradeable` contract to include the `whenNotPaused` modifier as well.

Bitcorn: Fixed in commit 1947399.

Cantina Managed: Verified.

3.2 Informational

3.2.1 Use namespaced storage layout for upgradeable contracts or immutables when possible

Severity: Informational

Context: AuthNoOwner.sol#L12-L13

Description: `AuthNoOwner` is initializable with the `_authority` and uses regular storage slot layout. All other inherited contracts of CornOFT use ERC7201 namespaced storage layout. Care should be taken that during an upgrade this slot layout is respected in the new version.

Recommendation: Consider either switching to a namespaced storage layout or if possible make `authority` an `immutable` set in the constructor thereby removing the need for the `_authorityInitialized`.

Bitcorn: Thank you. We'll take the suggestion into account when upgrading the contract.

Cantina Managed: Acknowledged.

3.2.2 `OFTVotes` can be simplified to avoid code duplication

Severity: Informational

Context: OFTVotes.sol#L13

Description: The `OFTVotes` contract can be simplified by inheriting from `OFTUpgradeable` instead of `OFTCoreUpgradeable`. Currently, it defines four functions relevant to cross-chain transfers, including `token()`, `approvalRequired()`, `_debit()`, and `_credit()`, all of which are the same as defined in the `OFTUpgradeable` contract (see OFTUpgradeable.sol). Inheriting from `OFTUpgradeable` directly would simplify the codebase.

Recommendation: For the `OFTVotes` contract, consider inheriting from `OFTUpgradeable` instead of `OFTCoreUpgradeable`. If there is a need for custom implementation of any of the `OFTUpgradeable` functions, consider overriding them in the `OFTVotes` contract.

If the `OFTVotes` contract remains the same, consider removing the `_localDecimals` parameter in the constructor and changing `OFTCoreUpgradeable(_localDecimals, _endpoint)` to `OFTVotes(decimals(), _lzEndpoint)`. As `_localDecimals` should be the decimals of the OFT (18 decimals by default), it ensures no mismatch between `_localDecimals` and `decimals()`.

Bitcorn: Fixed in PR 35.

Cantina Managed: Verified.