CANTINA

Impossible Cloud Network Protocol & Link Smart Contracts

Security Review

Cantina Managed review by:

Joran Honig, Lead Security Researcher
Hash, Security Researcher

October 7, 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

ICN builds the first composable, permissionless and truly open cloud ecosystem, this without compromising performance.

From Aug 24th to Aug 29th the Cantina team conducted a review of icn-protocol and icn-link-smart-contract on commit hashes 7506e28e and deff5e3d respectively. The team identified a total of **10** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	1	1	0
Medium Risk	2	2	0
Low Risk	3	3	0
Gas Optimizations	1	1	0
Informational	3	1	2
Total	**10**	**8**	**2**

The Cantina Managed team reviewed Impossible Cloud's `icn-protocol` and `icn-link-smart-contract` holistically on commit hashes c93ab217 and d3cbae92 respectively.

In addition, the Cantina Managed team reviewed the following set of changes on top of the findings fixes:

ICN Link Smart Contract:

PR	Description	Cantina Managed Team's Response	Impossible Cloud's Response
PR 81	Remove unused events and deprecate variable	Fix verified. Removes unused code.	-
PR 82	Fix natspec comments	Fix verified. No code changes.	-
PR 83	Remove `decreaseTime` function	Fix verified. Removes unused code.	-

ICN Protocol:

PR	Description	Cantina Managed Team's Response	Impossible Cloud's Response
PR 279	Merge main (migration) to release branch	Only reviewed `src/modules/Migrations/Migration2.sol`. The migration can be called multiple times which can cause undefined behaviour. The migration can be frontran causing it to misbehave (claiming one claim will change which index points to what claim). As a result you might migrate the incorrect claims, or the transaction could fail if the relevant claim is not present.	Acknowledged, this module has been removed and will not be used anymore. The function was called with a previous data check script but the check should have been done in function.

PR 282	Fix breaking changes / standardise deprecation comment / rename multi to batch function	Change to HW class return value are reverted in another pr. Documentation for multi claim rewards in interface is not updated. The function name change is not backwards compatible, and external contracts might depend multiClaimAwards being available. Though, this might be fine if you're sure there are no external integrations that depend on this interface.	Right, it was an error in the first place. We still want to call this function multi but for consistency for now we named it batch. We have a ticket to rename all function that do multicalls. So documentation remain correct even if not matching function name. This function is new to this release and still unused externally.
PR 285	Add V2 event for unindexed array output	What's the reason for this change? **Update after Impossible Cloud's response**: This does add a bit of complexity on the smart contract side to support off chain simplicity. That's discretionary though.	Indexer was needing array output unindexed
PR 287	Fix unfixed breaking change	Reverts changes to HW class interface from PR 282. This change requires an accompanying migration as new protocol margin storage is not initialized. Removing a variable from a struct in storage is dangerous, as future extentions of the struct will re-use that storage which is potentially populated. This can lead to some unfortunate vulnerabilities. What is the reason for changing from `calldata` to `memory`? for `_setProtocolMargin`.	Correct. The new protocol margin is initialized for all current region/hwClass in initializeICNRegistry and then set for each registration of new region/hwClass. This parameter protocolMargin was added to struct in this release. So removing it from HWClass struct is just reverting the changes back to current version. Goal was to avoid introducing breaking changes on registerRegion input params, which is calling through `_registerHwClass` the function `_setProtocolMargin`
PR 288	Add EraManager v3 initializer	Fix verified. Only code change is a version bump.	-

After the holistic review of the aforementioned PRs, it was concluded that all findings were addressed and no new vulnerabilities were identified.

3 Findings

3.1 High Risk

3.1.1 ScalerNode can be removed before its utilized capacity is reset

Severity: High Risk

Context: BookingManager.sol#L174-L201, ICNRegistry.sol#L419-L467

Description: Users can create bookings for resources on ScalerNodes. The `BookingManager` is responsible for updating the utilized capacity trackers for the region, cluster and hardware provider for the node that's being booked/re-booked. As a booking expires the updates to these trackers are not updated immediately, instead a call to `expireCapacity()` needs to be made which will perform the desired operations.

Currently a node can be removed by a call to `removeScalerNode()` on `ICNRegistry` which does not check whether the capacity for the node is still marked as utilized. After removing the node it will be impossible to revert the utilized capacity updates that were made in creating the last booking for the removed node.

As a result various computations around protocol rewards will be inaccurate since they'll operate with inaccurate utilized capacity numbers.

Recommendation: Adjust the implementation of `removeScalerNode()` with a check that ensures a node can only be removed if there is no booking that still needs to be expired.

ICN Protocol: Fixed in PR 275.

Cantina Managed: Fix verified.

3.2 Medium Risk

3.2.1 Users are not guarded against price increase in the `extendBooking` path

Severity: Medium Risk

Context: BookingManager.sol#L149-L150

Description: Users are guarded against unexpected price increases in the `bookCapacity` function by specifying a maxBookingPrice. But this guard is not present in the `extendBooking` function where similar similar issue can occur.

Recommendation: Add a similar guard in `extendBooking`.

ICN Protocol: Fixed in PR 274.

Cantina Managed: Fix verified.

3.2.2 Link token ids must be smaller than the max uint32 value

Severity: Medium Risk

Context: LinkStaking.sol#L330-L334

Description: The modules supporting link staking and reward claiming assume that link token ids have a value lower than the max uint32 value. Currently the ICN link contract supports minting tokens that do not satisfy this constraint, as a result governance might accidentally mint tokens that are incompatible with the modules.

Recommendation: Introduce a require check that ensures that _mint only succeeds for tokens that have valid token ids.

ICN Protocol: Fixed in PR 80.

Cantina Managed: Fix verified.

3.3 Low Risk

3.3.1 Checkpoints may be increased even when reward amount is 0 due to rounding

Severity: Low Risk

Context: HPDelegationICNT.sol#L694-L696

Description: The `batchInitiateDelegationRewardsClaim` function only checks that the sum of `_unclaimedRewards` has to be non-zero. This allows individual claims to have 0 as the reward amount due to rounding in which case the checkpoints are still incremented. This can cause users to lose dust amount of rewards continuously.

Recommendation: Revert in case a single `_claimAmount` is 0.

ICN Protocol: Fixed in PR 271.

Cantina Managed: Fix verified.

3.3.2 Incorrect index is emitted in `DelegationCreated` event

Severity: Low Risk

Context: HPDelegationICNT.sol#L662-L670

Description: The `DelegationCreated` event is supposed to emit the index of the locked delegation rather than the node delegation's index. But the implementation currently emits the index of the nodeDelegation.

```
/// @notice Emitted when a delegation is created.
    /// @param nodeId The ID of the node
    /// @param delegator The address of the delegator
    /// @param lockedDelegationIndex The index of the locked delegation
    /// @param apyScalingFactor The APY scaling factor for the delegation
    /// @param unlockTimestamp The timestamp when the collateral can be unlocked
    /// @param amount The amount of collateral delegated
    event DelegationCreated(
        uint256 indexed nodeId,
        address indexed delegator,
        uint256 indexed lockedDelegationIndex,
        uint256 apyScalingFactor,
        uint256 unlockTimestamp,
        uint256 amount
    );
```

Recommendation: Emit the UserDelegation index.

ICN Protocol: Fixed in PR 268.

Cantina Managed: Fix verified.

3.3.3 The `store32` library function should ensure input validity

Severity: Low Risk

Context: ArraysLib.sol#L8-L40

Description: The `store32()` function is used to efficiently store an array of integers in storage. There is a notable difference between the input type and output type. Namely the storage array is for `uint32` and the input array contains `uint256`. It functions correctly if all numbers in the input array are smaller than `type(uint32).max`. However, the functions' behavior becomes undefined if the input has larger integers. Since the parameter types allow `uint256` the function should function under all potential values of that type.

Recommendation: Introducing a require ensuring the input values are lower than the maximum `uint32` value or changing the input type to an array of `uint32` would ensure the function has no undefined or inaccurate behavior.

ICN Protocol: Fixed in PR 273.

Cantina Managed: Fix verified.

3.4 Gas Optimization

3.4.1 Node removal always swaps nodes

Severity: Gas Optimization

Context: ICNRegistry.sol#L450-L453

Description/Recommendation: Swapping the final ScalerNode with the to-be-deleted node should only be necessary when the to be deleted node is not the last node in the array.

ICN Protocol: Fixed in PR 269.

Cantina Managed: Fix verified.

3.5 Informational

3.5.1 Limited domain isolation between modules

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The architecture of ICN is aimed at having a single upgradable proxy contract that has multiple modules responsible for parts of the protocol. As expected there are some dependencies between modules. For example, the `BookingManager` needs to know if a node is bookable and will depend on the `ICNRegistry` module for that.

Each module is accompanied by a storage contract which provides an interface to the storage domain for that module. Currently most cross-module operations will perform direct look-ups and mutations to the storage of other modules.

This creates a situation where the responsibilities and dependencies between modules can be more opaque. For example, the `BookingManager` is responsible for recording the utilized capacity of resources otherwise managed by the `ICNRegistry`. In bug #1 we see that this leads to a potential misaccounting of capacities since `ICNRegistry` does not account for utilized capacity.

Recommendation: It is worth exploring the introduction of internal functions to storage contracts that wrap external reads and writes to make cross-module dependencies explicit. This will not prevent bugs but can help with code understanding.

ICN Protocol: Acknowledged.

Cantina Managed: Acknowledged.

3.5.2 Inconsistency in the reward formula b/w implementation and documentation

Severity: Informational

Context: Node Capacity Rewards.md

Description: According to the documentation rewards should be distributed as.

$$R_{base}(n_i, t) = \frac{x_{n_i}}{\sum_{n_i \in C} x_{n_i}(t)}(t) c_G(t) u_G(t) b_G(t)$$

where $x_{n_i}(t)$ is the total capacity of the node n_i at the time t.

But the implementation uses `targetCapacity` in the denominator rather than $\sum x_{n_i}(t)$.

Recommendation: Correct the documentation.

ICN Protocol: Fixed in PR 270.

Cantina Managed: Fix verified.

3.5.3 Introduce Fuzz Testing and Testing Enhancements

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The ICN codebase has several aspects that are quite amenable to fuzz testing and (bounded) formal verification.

The following are some example areas:

- Bitmap implementation used for efficient batch staking and initial rewards computation.
- Complex arithmetic operations.
- Library implementation for array copies.

Testability: To make introducing fuzz testing easier it can help to isolate (more) pure logic in libraries. This also aids in building a regular test suite and scaling the codebase as it grows in complexity.

For example, the logic that structures the link initial reward bitmap and the link staking bitmap implements is roughly equivalent and generalizable. Though the operations are relatively straightforward, it is nice to be able to test such low level operations directly. Implementing a bitmap library that efficiently supports single & batch operations can greatly simplify the link modules while simultaneously allowing for extensive unit tests and techniques such as fuzzing.

Fuzzing invariants: The areas of interest noted above can be tested in a stateless fashion, this is generally easier both on the developer and the fuzzing tool. The following are example invariants that you might consider introducing fuzz tests for:

1. Given two differing id's in the bitmap domain (e.g. link ids):
    1. Marking the first will only work if it is not marked.
    2. Marking only the first will not set the second as marked.
2. Given any list of integers the `store32` function will populate a list in storage where for every index the original array and storage array will have the same value.

Stateful fuzzing: There is also an opportunity for stateful fuzzing. For example bug #1 might have been discovered by the following invariant:

> The utilized capacity for a region/hp/cluster is equal to the sum of recorded utilized capacity of those nodes that belong to the region/hp/cluster.

Similar invariants might be formulated for other aspects of cross module accounting. Note that stateful fuzzing requires a more complex setup, so we recommend focussing on the invariants above first.

ICN Protocol: Acknowledged. This task is now part of our backlog.

Cantina Managed: Acknowledged.