Infrared contracts

Security Review

Cantina Managed review by:

MiloTruck, Lead Security Researcher
Cryptara, Security Researcher

March 23, 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

Infrared simplifies interacting with Proof of Liquidity with liquid staking products such as iBGT and iBERA.

From Mar 20th to Mar 21st the Cantina team conducted a review of infrared-contracts on commit hash 127c9f83. 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	0	0	0
Gas Optimizations	0	0	0
Informational	3	1	2
Total	**3**	**1**	**2**

3 Findings

3.1 Informational

3.1.1 Minor improvements to code

Severity: Informational

Context: *See each case below*

Description/Recommendation:

1. InfraredV1_3.sol#L11-L12 -- These imports, including `ConfigTypes`, are unused and can be removed.
2. InfraredV1_3.sol#L77-L86 -- The functionality in `queueMultipleValCommissions()` is a duplicate of `queueValCommission()`, consider having an internal function to avoid repeated code and to save some deployment gas:

```
function queueValCommission(bytes calldata _pubkey, uint96 _commissionRate)
    external
    onlyGovernor
{
    _queueValCommission(_pubkey, _commissionRate);
}

function queueMultipleValCommissions(
    bytes[] calldata _pubkeys,
    uint96 _commissionRate
) external onlyGovernor {
    uint256 length = _pubkeys.length;
    for (uint256 i = 0; i < length; i++) {
        _queueValCommission(_pubkeys[i], _commissionRate);
    }
}

function _queueValCommission(bytes calldata _pubkey, uint96 _commissionRate) internal {
    if (!isInfraredValidator(_pubkey)) revert Errors.InvalidValidator();
    IBeraChef(address(chef)).queueValCommission(_pubkey, _commissionRate);
    emit ValidatorCommissionQueued(msg.sender, _pubkey, _commissionRate);
}
```

Infrared: Fixed in commit 6717cd8.

Cantina Managed: Verified.

3.1.2 Observations regarding `queueValCommission()`

Severity: Informational

Context: InfraredV1_3.sol#L57-L64.

Description: In `BeraChef`, if `queueValCommission()` is called for `valPubKey` where there is already a queued commission change, the function will revert:

```
/// @inheritdoc IBeraChef
function queueValCommission(bytes calldata valPubkey, uint96 commissionRate) external onlyOperator(valPubkey) {
    if (commissionRate > ONE_HUNDRED_PERCENT) {
        InvalidCommissionValue.selector.revertWith();
    }
    QueuedCommissionRateChange storage qcr = valQueuedCommission[valPubkey];
    if (qcr.blockNumberLast > 0) {
        CommissionChangeAlreadyQueued.selector.revertWith();
    }
    (qcr.blockNumberLast, qcr.commissionRate) = (uint32(block.number), commissionRate);
    emit QueuedValCommission(valPubkey, commissionRate);
}
```

This impacts the `InfraredV1_3` contract in two ways:

1. If `queueMultipleValCommissions()` is called with duplicate public keys in the `_pubKeys` array, the function will revert. This is intended behavior.

2. If the governor wants to change a validator's commission rate while an update is queued (i.e. after `queueValCommission()` is called), he cannot do it immediately. He has to wait for the commission delay to end and before calling `activateQueuedValCommission()` -> `queueValCommission()` again. This is a limitation imposed by the `BeraChef` contract and cannot be changed.

Recommendation: Document the behavior mentioned in point (2).

Infrared: Acknowledged.

Cantina Managed: Acknowledged.

3.1.3 `InfraredV1_2.addValidators()` can be overridden to set a validator's commision rate

Severity: Informational

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

Context: InfraredV1_2.sol#L710-L723.

Description/Recommendation: Since the default `_commisionRate` for validators is `1e4` (i.e. 100%), consider overriding the `addValidators()` function from `InfraredV1_2` and directly calling `BeraChef.queueValCommission()` whenever a validator is added.

Note that any call to `queueValCommission()` triggers a waiting period defined by `commissionChangeDelay`. Therefore, setting the commission rate during `addValidators()` will not immediately result in the validator having a 100% commission.

Infrared: Acknowledged.

Cantina Managed: Acknowledged.