CANTINA

Telcoin: tel-v3-staking

Security Review

Cantina Managed review by:
Kaden, Lead Security Researcher

July 23, 2026

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**
- 2.1 Scope . . . 3

3 Findings **4**
- 3.1 Low Risk . . . 4
  - 3.1.1 `claimable` and `claimableAt` do not deduplicate epoch ids and can over-report the claimable total . . . 4
  - 3.1.2 Owner can pause a plugin through its deactivation window to strand user rewards . . . 4
  - 3.1.3 `increaseClaimableBy` is missing an `address(0)` guard . . . 5
  - 3.1.4 Streaming rewards use a single-block snapshot rather than staking duration, enabling reward gaming . . . 5
  - 3.1.5 Batch migrator can migrate any user's V2 stake without their authorization . . . 6
  - 3.1.6 `migrateAll` stakes claimed V2 rewards as principal when `v2AuxData` claims rewards 6
- 3.2 Gas Optimization . . . 6
  - 3.2.1 `_deactivationDelay` can be `immutable` . . . 6
  - 3.2.2 Redundant arithmetic in the `setWithdrawalDelay` rate-limit check . . . 7
- 3.3 Informational . . . 7
  - 3.3.1 Constructor parameter `_deactivationDelay` shadows the inherited storage variable 7
  - 3.3.2 `DEFAULT_ADMIN_ROLE` is the top of the trust hierarchy . . . 8
  - 3.3.3 Incorrect NatSpec `@title` tag on `DeactivationTimelock` . . . 8
  - 3.3.4 `updateRewardToken` does not emit `TokenUpdated` . . . 8
  - 3.3.5 `initialize` does not emit `WithdrawalDelaySet` for the initial delay . . . 9

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

Telcoin creates low-cost, high-quality financial products for every mobile phone user in the world.

From Jul 9th to Jul 17th the Cantina team conducted a review of tel-v3-staking on commit hash 587bb3ed. The team identified a total of **13** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	6	5	1
Gas Optimizations	2	2	0
Informational	5	5	0
Total	**13**	**12**	**1**

2.1 Scope

The security review had the following components in scope for tel-v3-staking on commit hash 587bb3ed:

```
src
├── SimplePlugin.sol
├── StakingModule.sol
├── interfaces
│   ├── IPlugin.sol
│   ├── ISimplePlugin.sol
│   └── IStakingModule.sol
├── migration
│   ├── StakingMigratorV2toV3.sol
│   └── interfaces
│       ├── IStakingModuleV2.sol
│       └── ITokenMigration.sol
└── utils
    └── DeactivationTimelock.sol
```

3 Findings

3.1 Low Risk

3.1.1 `claimable` and `claimableAt` do not deduplicate epoch ids and can over-report the claimable total

Severity: Low Risk.

Context: StreamingRewardsPlugin.sol#L264-L276.

Description: `claimable()` and `claimableAt()` sum an account's payout over a list of `epochIds` decoded from `auxData`, without checking that the ids are unique. Being `view` functions, they never write the `_claimedAtBlock` ledger, so a repeated id passes the `_claimedAtBlock[...] != 0` guard on every iteration and its `_grossPayout(...)` is added again. For example, a caller passing `[5, 5, 5]` receives three times epoch 5's payout.

While no in-scope contract currently consumes these functions, if a downstream contract that treats the returned total as authoritative is introduced in the future, it could act on an arbitrarily inflated amount.

Recommendation: Consider either documenting that the total is only valid for a duplicate-free list, or enforcing uniqueness by requiring the caller to pass `epochIds` pre-sorted in strictly increasing order. The latter approach can be validated with the following snippet inside the `for` loop:

```
if (i != 0 && epochId <= prev) revert UnsortedEpochIds(epochId, prev);
prev = epochId;
```

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.

3.1.2 Owner can pause a plugin through its deactivation window to strand user rewards

Severity: Low Risk.

Context: SimplePlugin.sol#L438-L445.

Description: The deactivation window exists to give users time to claim outstanding rewards before a plugin goes dark, as stated in the `startDeactivation()` docs. This protection can be bypassed by the owner. In both `SimplePlugin` and `StreamingRewardsPlugin`, `claim()` and `increaseClaimableBy()` are gated by `whenNotPaused`, but `_deactivated()` advances purely on `block.timestamp` and `cleanupPostDeactivation()` is gated only by `whenDeactivated`, not `whenNotPaused`.

As a result, the owner can call `startDeactivation()` and then `pause()`. The deactivation timer keeps running while the contract is paused, so `claim()` reverts for the entire window. Once the window closes, the owner calls `cleanupPostDeactivation()`, which still executes while paused and sweeps all remaining reward funds to the owner. Users with outstanding rewards receive nothing.

The ability to pause is a legitimate response to a security incident, so this may be an acceptable tradeoff. It is worth being aware of, as the protection the deactivation window is meant to provide ultimately depends on the trust level of the plugin owner.

Recommendation: Consider freezing the deactivation timer while the contract is paused.

Telcoin Association: Acknowledged.

Cantina Managed: Acknowledged.

3.1.3 `increaseClaimableBy` is missing an `address(0)` guard

Severity: Low Risk.

Context: SimplePlugin.sol#L265.

Description: The single-credit `increaseClaimableBy()` has no `account != address(0)` check, while the batch path `increaseClaimableByBatch()` rejects a zero address with `ZeroAddress`. Calling `increaseClaimableBy(address(0), amount)` pulls `amount` reward tokens into the plugin, credits `_claimable[address(0)]`, and adds `amount` to `_totalOwed`.

`StakingModule` always claims on behalf of `msg.sender`, so `address(0)` can never claim, and the credited funds are permanently locked. `totalClaimable()` stays overstated by `amount`, and because that amount is counted in `_totalOwed` it is treated as owed rather than free, so `rescueTokens()` cannot recover it either.

Recommendation: Consider adding the same guard used by the batch path:

```
+ if (account == address(0)) revert ZeroAddress();
```

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.

3.1.4 Streaming rewards use a single-block snapshot rather than staking duration, enabling reward gaming

Severity: Low Risk.

Context: StreamingRewardsPlugin.sol#L37-L41.

Description: Streaming rewards pay `rewardAmount * getPastVotes(account, refBlock) / totalSupplyAtRef`, a single-block snapshot taken at `refBlock`. An account's reward is proportional to its stake at one block per cycle, with no weighting by how long the stake was actually held. The `refBlock < block.number` rule prevents front-running a snapshot that is already known, but it does not prevent positioning for a predictable future snapshot.

This design raises three concerns:

1. Timing predictability. Epochs are expected to be funded frequently, but the cadence must not be predictable. If `refBlock` lands at the same time each day, or even within a predictable window, a user can stake only around the anticipated block and unstake afterward, capturing a full proportional share for minimal staking duration.
2. Fairness. Because only the snapshot counts, rewards track when a user happens to be staked at the checkpoint rather than the total time staked. Two stakers holding the same amount are rewarded very differently based purely on timing.
3. Distributor trust. The `distributor` selects `refBlock`, so it could time epochs around its own stake to disproportionately reward itself.

The withdrawal delay partially mitigates the first two by keeping capital locked, with effectiveness depending on the delay length relative to epoch frequency, but it does not address the underlying design.

Recommendation: Consider a design that rewards stakers based on the duration staked rather than presence at discrete checkpoints. One approach is to give epochs a start and end time, ideally contiguous, and on each claim or stake/unstake sum the rewards for every full epoch the account was staked through, then pro-rate partially staked epochs by the time staked within them. The same effect can be achieved without epochs by accruing a constant reward rate per unit of stake per unit of time (a reward-per-token accumulator), and the rate itself can be made a function of time if a variable schedule is desired.

If the snapshot design is retained, ensure the epoch cadence and `refBlock` selection are unpredictable and jittered, and do not run epochs while the withdrawal delay is `0`.

Telcoin Association: Fixed in PR 41.

Cantina Managed: Fix verified.

3.1.5 Batch migrator can migrate any user's V2 stake without their authorization

Severity: Low Risk.

Context: StakingMigratorV2toV3.sol#L336-L351.

Description: `migrateBatch()`, gated by `BATCH_MIGRATOR_ROLE`, exits each listed user's V2 stake through `claimAndExitFor()`, swaps it to V3, and re-stakes it to the user via `stakeForBatch()`. No approval or signature is required from the users being migrated. Each user receives their equivalent V3 stake, so there is no loss, but a user has no say in whether or when they are migrated.

This matters if some users would prefer not to migrate, for example because their V2 stake is still earning rewards, they want to remain liquid, or they do not want to be subject to the V3 withdrawal delay. The design therefore depends on the `BATCH_MIGRATOR_ROLE` holder being fully trusted to migrate only users who have authorized it off-chain.

Recommendation: If forced migration is not intended, consider requiring each user to sign a message authorizing their own migration and validating that signature in `migrateBatch()` before exiting their V2 stake. If forced migration is intended, consider documenting this trust assumption clearly so users understand their stake can be moved without an on-chain action on their part.

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.

3.1.6 `migrateAll` stakes claimed V2 rewards as principal when `v2AuxData` claims rewards

Severity: Low Risk.

Context: StakingMigratorV2toV3.sol#L285-L287.

Description: `migrateAll()` is documented to preserve each caller's staked versus unstaked split by re-staking only the previously staked portion and returning the previously unstaked portion to the wallet. However, there is one exception to this pattern.

Leg 1 calls `claimAndExitFor()`, which pays both the exited staked principal and any plugin rewards claimed through `v2AuxData` to the migrator. `stakedPulled` is measured as the aggregate balance delta, so it includes those rewards. Leg 3 then stakes the whole amount as sTEL, so reward tokens that were not staked before migration end up staked. The `MigratedAll` event's `stakedPulled` and `stakedV3` fields likewise over-report principal. Because the resulting sTEL value is equivalent, no funds are lost. Regardless, the actual behavior differs from the documented behavior.

Recommendation: Consider either measuring the exited principal and the claimed rewards separately, so the principal can be re-staked while the rewards are returned to the wallet as unstaked, or documenting that rewards claimed through `v2AuxData` are folded into the staked portion.

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified. The behavioral discrepancy is addressed at the UI level: rewards claimed via `v2AuxData` during migration are folded into the re-staked principal, and this is documented in the UI rather than changed in code.

3.2 Gas Optimization

3.2.1 `_deactivationDelay` can be `immutable`

Severity: Gas Optimization.

Context: DeactivationTimelock.sol#L13-L14.

Description: `_deactivationDelay` in `DeactivationTimelock` is assigned only in the constructor and is never modified afterward. Both `SimplePlugin` and `StreamingRewardsPlugin` pass a fixed value into that constructor and are deployed directly rather than behind a proxy, so the value is set once at deployment.

Declaring it `immutable` stores it in contract bytecode rather than storage, replacing the storage load in `_startDeactivation()` and the public getter with a cheaper bytecode read.

Recommendation: Consider marking the variable `immutable`:

```
- uint256 public _deactivationDelay;
+ uint256 public immutable _deactivationDelay;
```

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.

3.2.2 Redundant arithmetic in the `setWithdrawalDelay` rate-limit check

Severity: Gas Optimization.

Context: StakingModule.sol#L764-L766.

Description: In the increase branch of `setWithdrawalDelay()`, the per-day cap is computed as `(elapsed * MAX_DELAY_INCREASE_PER_DAY) / 1 days`. Since `MAX_DELAY_INCREASE_PER_DAY` is set to `1 days`, this expression is always equal to `elapsed`, so the multiplication and division are redundant. The compiler does not fold them because it must preserve the overflow behavior of the intermediate `elapsed * 1 days`.

Recommendation: Consider comparing against `elapsed` directly:

```
- uint256 maxIncrease = (elapsed * MAX_DELAY_INCREASE_PER_DAY) / 1 days;
  uint256 requested = newDelay - oldDelay;
- if (requested > maxIncrease) revert DelayIncreaseTooFast(requested, maxIncrease);
+ if (requested > elapsed) revert DelayIncreaseTooFast(requested, elapsed);
```

Note that this couples the check to `MAX_DELAY_INCREASE_PER_DAY` being equal to `1 days`. If that constant is ever changed, the simplification must be revisited.

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.

3.3 Informational

3.3.1 Constructor parameter `_deactivationDelay` shadows the inherited storage variable

Severity: Informational.

Context: StreamingRewardsPlugin.sol#L206.

Description: The `StreamingRewardsPlugin` constructor declares a parameter named `_deactivationDelay`, which shares its name with the storage variable inherited from `DeactivationTimelock`. The constructor uses the locally scoped parameter as intended, so behavior is correct, but the shared name makes it harder to distinguish the parameter from the storage variable.

Recommendation: Consider renaming the parameter, for example to `deactivationDelay_`, and updating its references in the constructor:

```
- constructor(address _staking, address _rewardToken, uint256 _deactivationDelay)
+ constructor(address _staking, address _rewardToken, uint256 deactivationDelay_)
      Ownable(msg.sender)
-     DeactivationTimelock(_deactivationDelay)
+     DeactivationTimelock(deactivationDelay_)
```

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.

3.3.2 `DEFAULT_ADMIN_ROLE` is the top of the trust hierarchy

Severity: Informational.

Context: StakingModule.sol#L270-L276.

Description: The `_authorizeUpgrade()` documentation describes `UPGRADER_ROLE` as the top of the trust pyramid, protected by a multisig and timelock so any malicious upgrade has a public window during which users can exit. Roles are managed with OpenZeppelin `AccessControl`, where `DEFAULT_ADMIN_ROLE` is the admin of every role, including `UPGRADER_ROLE` and itself.

A `DEFAULT_ADMIN_ROLE` holder can therefore grant `UPGRADER_ROLE` to any address, including itself, and revoke or reassign roles at will, which places it above `UPGRADER_ROLE`. If `UPGRADER_ROLE` is guarded by a timelock but `DEFAULT_ADMIN_ROLE` is not, an admin could grant itself `UPGRADER_ROLE` and upgrade without the intended public window, bypassing the protection.

Recommendation: Consider ensuring `DEFAULT_ADMIN_ROLE` is protected at least as strongly as `UPGRADER_ROLE`, for example a multisig combined with a timelock.

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.

3.3.3 Incorrect NatSpec `@title` tag on `DeactivationTimelock`

Severity: Informational.

Context: DeactivationTimelock.sol#L4-L9.

Description: The NatSpec block above `DeactivationTimelock` uses `@title TieredOwnership`, which names a different contract and appears to have been copied from `TieredOwnership.sol`. The title does not describe the contract it sits above.

Recommendation: Update the `@title` tag to match the contract it documents.

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.

3.3.4 `updateRewardToken` does not emit `TokenUpdated`

Severity: Informational.

Context: FeeBuybackProxySafe.sol#L148-L152.

Description: `updateRewardToken()` in `FeeBuybackProxySafe` changes the `telcoin` reward token but does not emit an event. The `TokenUpdated` event already exists and is emitted by `initialize()` when the token is first set, so the update path is inconsistent and leaves offchain monitoring unaware of any later change.

Recommendation: Consider emitting `TokenUpdated` after the assignment:

```
  function updateRewardToken(
      IERC20 token
  ) public onlyRole(SUPPORT_ROLE) {
      telcoin = token;
+     emit TokenUpdated(address(telcoin));
  }
```

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.

3.3.5 `initialize` does not emit `WithdrawalDelaySet` for the initial delay

Severity: Informational.

Context: StakingModule.sol#L218.

Description: `initialize()` sets `$.withdrawalDelay` to `initialWithdrawalDelay_` but does not emit an event. The `WithdrawalDelaySet` event already exists and is emitted by `setWithdrawalDelay()` on every later change, so the initial value is the only one not surfaced through an event.

Recommendation: Consider emitting `WithdrawalDelaySet` from the old value of `0` to the initial delay after the assignment:

```
  $.withdrawalDelay = initialWithdrawalDelay_;
+ emit WithdrawalDelaySet(0, initialWithdrawalDelay_);
```

Telcoin Association: Fixed in PR 26.

Cantina Managed: Fix verified.