Tinyman Liquid Staking Audit

Appendix: Summary of Findings and Responses

Below is a deduplicated list of the findings from the review, along with our responses and remediations where applicable.

---

tAlgo Application

1. Unbounded Protocol Fee Rewards

Issue:
The manager can set the protocol fee to a value greater than 100%, causing two potential issues:

- Users' staked Algo can be reallocated as protocol fees, causing users to lose both principal and rewards.
- An excessively high fee percentage can lead to overflows in calculations, causing mint and burn functions to break and effectively block users from accessing their funds.

Response:
This issue is valid and arises from a lack of input validation in the `set_protocol_fee` function. The issue has been addressed by ensuring the fee percentage is always less than or equal to 100. We opted for this mathematically correct limit rather than a lower value to avoid setting incorrect expectations for future changes to this parameter.

Although this issue could only have been exploited by a malicious manager, one of the system's core design principles is that a malicious or careless manager should never be able to adversely affect user withdrawals or previously accrued rewards.

Remediation:
The issue was fixed in the following commit:
Fix commit

---

2. tAlgo Donations

Issue:
tAlgo transferred to the application account without the appropriate 'burn' function call results in locked Algo that cannot be claimed. It was suggested that these should be treated as 'donations' similar to Algo donations.

Response:
We agree with this observation. The code has been updated to recalculate the minted tAlgo amounts and rates based on the amounts held by the application account.

Remediation:
The issue was fixed in the following commits:
Commit 1
Commit 2

---

3. Key Registration Logic

Issue:
The system uses KeyReg transactions to register accounts online for participation in Algorand consensus. The application avoids inner transactions to allow flexibility for future changes. Initially, Logic Signatures were used for signing, but the mechanism was found to be unnecessarily complicated. It was suggested that signing authority could be temporarily transferred to the node manager.

Response:
We agreed with this assessment and have updated the system to use the node manager as the signer instead.

Remediation:
The change was implemented in the following commit:
Commit

The reviewers also raised concerns about the potential for dangerous fields being added to KeyReg transactions, which could create new attack vectors. We acknowledge this and trust the Algorand protocol developers and consensus participants to maintain security in future updates.

---

4. Setting the Manager Role

Issue:
The manager role has the authority to modify protocol settings (e.g., protocol fee, node managers, stake managers). However, the manager cannot rectify mistakes when updating the manager address, potentially locking the current manager out of the contract.

Response:
We agreed that this could be a risk and have implemented a two-step process where the current manager proposes a new manager. The manager role is updated only when the new manager accepts by submitting the appropriate application call.

Remediation:
The change was implemented in the following commit:
Commit

---

5. Code Optimizations

Issue:
The reviewers recommended various code optimizations to reduce code size, opcode costs, minimum balance requirements, and simplify logic.

Response:
We have incorporated these optimizations into the code as suggested.

---

6. Intentional Bug for Audit

Issue:
An intentional bug was included in the contracts to allow us to audit the audit process. The bug's hash was shared with reviewers at the start of the process, and all reviewers successfully identified the bug.

Remediation:
The fix for this issue was applied in the following commit:
Commit

Re-Staking Application

1. Manager Can Block Contract

Issue:
The manager can set the reward amount and duration through the `set_reward_rate` function. A malicious or careless manager could set an excessively high reward rate, causing an integer overflow in subsequent calculations. This would lock all funds permanently.

Response:
This issue is valid. We acknowledge that the manager role must not cause harm to principal or accrued rewards. We have fixed this by adding a check to ensure the reward rate does not cause overflows.

Remediation:
The fix was applied in the following commit:
[Commit]

---

2. Funds May Not Be Available for Rewards

Issue:
The application does not verify that sufficient funds are available to cover both unclaimed rewards and future rewards when the reward rate is updated. This could lead to a situation where users are unable to claim their earned rewards.

Response:
We have implemented a fix to ensure the required balance is checked before updating the reward rate.

Remediation:
The fix was applied in the following commit:
[Commit]

---

3. Insufficient TINY Power Checks

Issue:
The re-staking application checks TINY power only once when the user initially stakes their tAlgo. This allows users to continue re-staking without increasing their TINY power, even if the TINY power requirements increase.

Response:
We agree that the current implementation does not align with the intended feature. We have added an additional TINY power check during the claim step to ensure that users' TINY power remains sufficient for their staking activities.