Arable Staking

Smart Contract Audit Report

AUDIT SUMMARY

Arable Staking is releasing multiple ways to stake for rewards through various contracts.

For this audit, we reviewed the staking folder at commit 0e97457d835ff962172deada074023f3402f1e6d on the team's GitHub repository and the following addresses on the Avalanche mainnet:

- Farming contract at 0x598ebac38cf211749b1277c9a34d217226a476af.
- Staking contract at 0x4bc722cd3f7b29ae3a5e0a17a61b72ea5020502b.
- StakingRoot contract at 0x5c09b44ca09083cb3d62a05979f538a43359351c.

AUDIT FINDINGS

Please ensure trust in the team prior to investing as they have some control in the ecosystem.
Date: March 16th, 2022.
Updated: March 22nd, 2022 to include selected mainnet addresses and verify ownership was transferred.
Updated: March 23rd, 2022 to move from private to public GitHub repository.

Finding #1 - DStaking - Medium (Acknowledged)

Description: *The processRewards() function does not properly calculate the rewards claimed and the commision due when there is not sufficient tokens within the contract for a rewards claim.*
Risk/Impact: *The user who has claimed rewards will lose more rewards than they should and the team will receive more commission than they are due. This will lead to improperly distributed rewards over time.*
Recommendation: *The team should account for the two situations where the contract does or does not have sufficient tokens separately.*
Resolution: *The team has acknowledged this issue and plans to manually replenish rewards should they ever become insufficient.*

Finding #2 - TokenLocker - Low (Acknowledged)

Description: *The lockToken() function does not properly account for fee-on-transfer tokens.*
Risk/Impact: *When locking a fee-on-transfer token, users will be credited with locking more tokens than the contract actually receives. When withdrawing, they will withdraw the difference between the amount credited and actually received in other users' locked tokens.*
Recommendation: *The contract should record its original balance in tokens before initiating a transfer. After the transfer, users should be credited for the increase in tokens from the original recorded balance.*
Resolution: *The team has acknowledged this issue. The contract is not intended to be used with fee-on-transfer tokens within the platform, though users should exercise caution before locking fee-on-transfer tokens within the contract.*

Finding #3 - Farming - Informational (Resolved)

Description:*The owner of the StakingRoot contract may withdraw the entire contract balance of any token.*
Risk/Impact: *The owner is able to withdraw users' staked tokens.*

Recommendation: *The withdrawAnyToken() function should not be allowed to withdraw the designated staking token.*

Resolution: *The team has transferred ownership to a Gnosis Safe preventing abuse of this functionality.*

Finding #4 - Staking & DStaking - Informational (Resolved)

Description:*The owner of the StakingRoot contract may withdraw the entire contract balance of any token.*

Risk/Impact: *The owner is able to withdraw users' staked tokens.*

Recommendation: *If the withdrawAnyToken() function is called with the designated staking token address, the max withdrawable amount should be:*

```
uint256 amount = IERC20(_token).balanceOf(address(this)) - getTotalDepositAmount();
IERC20(_token).safeTransfer(beneficiary, amount);
```

Resolution: *The team has transferred ownership to a Gnosis Safe preventing abuse of this functionality.*

CONTRACTS OVERVIEW

- *The contracts utilize ReentrancyGuard to prevent against reentrancy attacks in applicable functions.*
- *As the contracts are implemented with Solidity v0.8.0, they are safe from any possible overflows/underflows.*
- *The team must exercise caution when assigning staking tokens to avoid using fee-on-transfer tokens; if a fee-on-transfer token is used, the staking contract should be exempt from transfer fees.*

Farming Contract:

- *Any user may deposit a specified staking token into farming pools to earn rewards in another specified token; each farming pool may have its own specified staking token and may vary between pools.*
- *Users will receive a reward amount on each block based on the amount staked and the amount of points allocated to the pool.*
- *Users may elect to claim their pending rewards on deposits and withdrawals.*
- *Users may also manually claim their rewards without a deposit or withdrawal.*
- *Users withdrawn tokens are locked in a TokenLocker contract for the pool's "lockup duration"; the lockup duration defaults to 14 days but may vary between pools.*
- *Tokens to be distributed as rewards must be supplied to the contract. If the contract does not have sufficient tokens, the remaining tokens in the contract will be distributed, preventing other users from claiming rewards until more tokens are supplied to the contract.*
- *The owner may withdraw any token from the contract at any time.*
- *The owner may pause the contract, disabling deposits, withdraws, and rewards claims, at any time.*
- *The owner may add a new pool at any time given that there is not already a pool with the same staking token and the pool's staking token is not the rewards token.*
- *The owner may update any pool's lockup duration and allocated points to any value at any time.*

FarmingFactory Contract:

- *This contract is used to create and track multiple Farming contracts.*
- *The owner may deploy a new Farming contract at any time.*
- *Ownership of the Farming contract is transferred to the owner upon deployment.*
- *The owner may update the TokenLocker contract used to lock withdrawals from the Farming contracts at any time.*

Staking Contract:

- *Any user may deposit specified tokens into a staking pool to earn rewards in the form of the same token.*
- *Users will receive a reward amount on each block based on the amount of tokens they have staked relative to the total amount staked in the pool.*
- *Users may elect to claim their pending rewards during withdrawals or may manually claim their rewards.*
- *Users withdrawn tokens are locked in a TokenLocker contract for the pool's "lockup duration" which defaults to 14 days.*
- *Tokens to be distributed as rewards must be supplied to the contract. If the contract does not have sufficient tokens, the remaining tokens in the contract will be distributed, preventing other users from claiming rewards until more tokens are supplied to the contract.*

- *The contract correctly calculates rewards seperately from the staked balance, preventing users from receiving staked tokens as rewards if not enough tokens are supplied to the contract.*
- *Any user may trigger a rewards claim from the StakingRoot contract at any time.*
- *The Root contract may withdraw any token from the contract at any time.*
- *The owner may update the specified token at any time. Doing so will reset the lockup duration to the default 14 days.*
- *The owner may pause the contract, disabling deposits, withdraws, and rewards claims, at any time.*
- *The owner may update the pool lockup duration to any value at any time.*

DStaking Contract:

- *Any user may "delegate" specified tokens into a DStaking pool to earn rewards, either for themself or on the behalf of another user, in the form of the same token.*
- *Users will receive a reward amount on each block based on the amount of tokens delegated to them relative to the total amount delegated within the pool.*
- *When enabled, users may redelegate their tokens once per "redelegation period"; redelegating will move tokens from one DStaking pool to another DStaking pool.*
- *Users may still claim rewards accumulated prior to redelegating.*
- *Users must manually claim their rewards; a commission fee is taken from all rewards.*
- *Tokens to be distributed as rewards must be supplied to the contract. If the contract does not have sufficient tokens, the remaining tokens in the contract will be distributed, preventing other users from claiming rewards.*
- *The contract correctly calculates rewards seperately from the delegate balance, preventing users from receiving staked tokens as rewards if not enough tokens are supplied to the contract.*
- *Users undelegated tokens are locked in a TokenLocker contract for the pool's "lockup duration" which defaults to 14 days.*
- *The contract creator may not withdraw more tokens than would put the contract's token balance below the minimum required in the StakingRoot contract.*
- *Any user may trigger a rewards claim from the StakingRoot contract at any time.*
- *The Root contract may update the pool lockup duration for withdrawn tokens to any value at any time.*
- *The Root contract may withdraw any token from the contract at any time.*
- *The owner may set the commission rate, if the minimum duration since the last update has passed, up to 20%.*
- *The owner may withdraw tokens collected from commission fees at any time.*

DStakingOverview:

- *This contract is used to track users' overall delegated balance.*
- *Users' balances are updated accordingly when altering their delegated token balance within any DStaking pool.*

StakingRoot Contract:

- *This contract is used to manage the DStaking and Staking contracts.*
- *Any approved DStaking creator addresses may create a new DStaking pool by supplying at least the minimum amount of specified tokens, which are subsequently delegated within the DStaking contract.*
- *The creator is given the Operator role and ownership of the DStaking contract.*
- *The creator of a DStaking contract or this contract's owner may remove a DStaking contract.*
- *When a DStaking contract is removed, rewards are distributed a final time and users may claim currently accumulated rewards.*
- *Any user may distribute rewards to the Staking and DStaking rewards pools at any time.*
- *The DStaking pools will receive rewards proportional to the total delegated tokens within the pool relative to the total staked and delegated tokens across all pools.*
- *The Staking pool will receive rewards proportional to the total staked tokens within the pool, multiplied by the "staking multiplier", relative to the total staked and delegated tokens across all pools.*
- *Any Staking or DStaking contract may claim rewards at any time; claiming rewards will transfer the tokens due as rewards to the claiming contract.*
- *Tokens to be distributed as rewards must be supplied to the contract. If the contract does not have sufficient tokens, the remaining tokens in the contract will be distributed, preventing other contracts from receiving rewards.*
- *The owner may pause the contract, preventing the ability to add DStaking contracts, remove DStaking contracts, and claim and distribute rewards, at any time.*
- *The owner may update the Staking, DStakingOverview, and TokenLocker contract addresses at any time.*

- *The owner may set the staking multiplier and the minimum amount of tokens needed to create a DStaking contract to any value at any time.*
- *The owner may withdraw any tokens from this contract and any valid Staking contract at any time.*
- *The owner may update the lockup duration for tokens withdrawn from DStaking contracts to any value at any time.*
- *The owner may add and remove any address from the approved DStaking creators list at any time.*
- *The owner may toggle DStaking redelegation at any time.*
- *The owner may set the redelegation period to any value at any time.*

TokenLocker Contract:

- *Any user may lock any token within this contract for a specified period of time.*
- *As any token can be deposited into the locker, it is possible for a malicious token to manipulate its balance within the contract. This will not have any effect on any other locked tokens.*
- *A beneficiary address that will receive the tokens when the lock is released is specified when locking tokens.*
- *The unlock time may be set to any time in the future.*
- *Any amount of tokens may be locked within the contract.*
- *There are no fees associated with locking or unlocking tokens.*
- *The beneficiary of the lock may release the lock once the unlock time has passed, receiving the locked tokens.*
- *Alternatively, any user may release a lock on the beneficiary's behalf given that the unlock time has passed and the recipient provided is the same as the beneficiary of the locked tokens.*

AUDIT RESULTS

Vulnerability Category	Notes	Result
Arbitrary Jump/Storage Write	N/A	PASS
Centralization of Control	• The team has transferred ownership to a Gnosis Safe contract. • The lock duration for withdrawn rewards is uncapped.	PASS
Compiler Issues	N/A	PASS
Delegate Call to Untrusted Contract	N/A	PASS
Dependence on Predictable Variables	N/A	PASS
Ether/Token Theft	N/A	PASS
Flash Loans	N/A	PASS
Front Running	N/A	PASS
Improper Events	N/A	PASS
Improper Authorization Scheme	N/A	PASS
Integer Over/Underflow	N/A	PASS
Logical Issues	• There are incorrect calculations in certain circumstances in the DStaking contract.	WARNING
Oracle Issues	N/A	PASS
Outdated Compiler Version	N/A	PASS

Vulnerability Category	Notes	Result
Race Conditions	N/A	PASS
Reentrancy	N/A	PASS
Signature Issues	N/A	PASS
Unbounded Loops	N/A	PASS
Unused Code	N/A	PASS
Overall Contract Safety		PASS

CONTRACT SOURCE SUMMARY AND VISUALIZATIONS

Name	Address/Source Code	Visualized (Hover-Zoom Recommended)
DStaking	GitHub	Inheritance Chart. Function Graph.
DStakingOverview	GitHub	Inheritance Chart. Function Graph.
Farming	GitHub	Inheritance Chart. Function Graph.
FarmingFactory	GitHub	Inheritance Chart. Function Graph.
Staking	GitHub	Inheritance Chart. Function Graph.
StakingRoot	GitHub	Inheritance Chart. Function Graph.
TokenLocker	GitHub	Inheritance Chart. Function Graph.

ABOUT SOLIDITY FINANCE

WHAT IS A SOLIDITY AUDIT?

Typically, a smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. A *Solidity Audit* takes this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members alike.

HOW DO I INTERPRET THE FINDINGS?

Each of our Findings will be labeled with a Severity level. We always recommend the team resolve High, Medium, and Low severity findings prior to deploying the code to the mainnet. Here is a breakdown on what each Severity level means for the project:

- **High** severity indicates that the issue puts a large number of users' funds at risk and has a high probability of exploitation, or the smart contract contains serious logical issues which can prevent the code from operating as intended.
- **Medium** severity issues are those which place at least some users' funds at risk and has a medium to high probability of exploitation.
- **Low** severity issues have a relatively minor risk association; these issues have a low probability of occurring or may have a minimal impact.
- **Informational** issues pose no immediate risk, but inform the project team of opportunities for gas optimizations and following smart contract security best practices.