SOLIDIFIED

Audit Report for SingularityNET - April 26, 2021

Summary

Audit Report prepared by Solidified covering the SingularityNET staking smart contract.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on April 26, 2021, and the results are presented here.

Audited Files

The source code has been supplied in a public source code repository:

https://github.com/singnet/snet-stake/tree/phase-2

Commit number: `b9c99546cfab9cc2177c242e6dbf42ccb321d67b`

Intended Behavior

The smart contract implements a contract that allows users to stake an ERC-20 token and earn rewards.

Findings

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

Issues Found

Solidified found that the SingularityNET staking contract contains no critical issue, no major issue, 1 minor issue, and 5 informational notes.

We recommend issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	TokenStake.sol: Operator can reject stakes from previous windows	Minor	Pending
2	TokenStake.sol: Pragma allows for a wide range of compiler versions	Note	-
3	TokenStake.sol: A staker is not removed from the stakeHolders array when they withdraw all their tokens or get rejected	Note	-
4	TokenStake.sol: Functions migrateStakes() and updateRewards() could be declared as external to save gas	Note	-
5	TokenStake.sol: Function depositToken() is redundant	Note	-
6	TokenStake.sol: Stake migration does not update total stake	Note	-

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. `TokenStake.sol`: Operator can reject stakes from previous windows

The operator can use any previous `stakeMapIndex` to reject a stake during the current stake submission end period. This is contrary to the documentation.

Recommendation
It is recommended to use the current stake index to validate the rejection.

Informational Notes

2. `TokenStake.sol`: Pragma allows for a wide range of compiler versions

Function `pragma` statement allows for a very large range of compiler versions, including some versions with known bugs. In addition, the language syntax has changed since the earlier versions that are allowed.

Recommendation
Consider limiting the compiler to at least a single major version number.

3. `TokenStake.sol`: A `staker` is not removed from the `stakeHolders` array when they withdraw all their tokens or get rejected

Upon withdrawing all their tokens via `withdrawStake()`, the `staker` will remain in the `stakeHolders` array even though they're no longer a stakeholder. The same issue could also occur if the `staker` is rejected in `rejectStake()`.

4. `TokenStake.sol`: Functions `migrateStakes()` and `updateRewards()` could be declared as `external` to save gas

Since both `migrateStakes()` and `updateRewards()` can potentially be passed large arrays, declaring them as `external` instead of `public` can save a significant amount of gas for the calling user.

5. `TokenStake.sol`: Function `depositToken()` is redundant

Function `depositToken()` is redundant, as any user is able to transfer `token` to the contract without calling it.

6. `TokenStake.sol`: Stake migration does not update total stake

The method migrateStakeWindow is not updating the `windowTotalStake` variable. Furthermore, this method can be used during an active stake.

Recommendation
Consider updating the `windowTotalStake` value with `windowRewardAmount`. Also, add validations to check if there is an existing stake in progress.

Disclaimer