Peer Review – SPL Stake Pool Redelegation Update

Neodyme AG

2023-01-31

Contents

Introduction

As part of the Solana peer review process, Neodyme was engaged to do a review of pull requests to the SPL stake view program. In particular, the following pull requests have been reviewed:

Title	Link
Support stake redelegation	https://github.com/solana-labs/solana-program-library/pull/3856
Add “IncreaseAdditionalValidatorStake” instruction	https://github.com/solana-labs/solana-program-library/pull/3924
Add DecreaseAdditionalValidatorStake instruction	https://github.com/solana-labs/solana-program-library/pull/3925

All reported issues have been fixed as of January 27th, 2023 (commit hash `b34102211f2a5ea6b83f3ee22f045fb115d87813`).
The program is still in active development, any changes past this point are out of scope for this report.

Project summary

The SPL stake pool program provides the ability for pooling together SOL to be staked by an off-chain agent running a Delegation Bot which redistributes the stakes across the network and tries to maximize censorship resistance and rewards.

SOL token holders can earn rewards and help secure the network by staking tokens to one or more validators. Rewards for staked tokens are based on the current inflation rate, total number of SOL staked on the network, and an individual validator’s uptime and commission (fee).

Stake pools are an alternative method of earning staking rewards. This on-chain program pools together SOL to be staked by a staker, allowing SOL holders to stake and earn rewards without managing stakes.

Contract expectations

Each user of the stake pool should be able to rely on the following two properties:

1. Safety: It is always possible to withdraw the stake deposited. The user should receive stake proportional to their pool share.
2. Fairness: Every user should receive the same relative rewards, so each user should only receive rewards proportional to their stake in the pool and not more.

Note that rewards are not guaranteed, as the manager of the stake pool can always decide to unstake the managed stake accounts. However, the “fairness” property ensures that assuming there is a well-behaved manager, all rewards will be distributed fairly among the users of the pool. The “safety” property ensures that if a user is no longer happy with the decisions of the manager, they can at any time decide to leave the pool and get back their share of stake.

Additionally, the manager should always receive the fees configured for possible user actions. There should be no way for any user to bypass the configured fees.

Methodology

Neodyme's audit team performed a comprehensive examination of the changes included in the aforementioned pull requests for the SPL stake pool program. The audit team, which consists of security engineers with extensive experience in Solana smart contract security, reviewed and tested the code, paying special attention to the following:

- Ruling out common classes of Solana contract vulnerabilities, such as:
  - Missing ownership checks
  - Missing signer checks
  - Signed invocation of unverified programs
  - Solana account confusions
  - Redeployment with cross-instance confusion
  - Missing freeze authority checks
  - Insufficient SPL account verification
  - Missing rent exemption assertion
  - Casting truncation
  - Arithmetic over- or underflows
  - Numerical precision errors
- Checking for unsafe design which might lead to common vulnerabilities being introduced in the future
- Checking for any other, as-of-yet unknown classes of vulnerabilities arising from the structure of the Solana blockchain
- Ensuring that the contract logic correctly implements the project specifications
- Examining the code in detail for contract-specific low-level vulnerabilities
- Ruling out denial of service attacks
- Ruling out economic attacks
- Checking for instructions that allow front-running or sandwiching attacks
- Checking for rug pull mechanisms or hidden backdoors

Scope

The audit encompassed the pull requests (patches) listed in “Introduction” (PR numbers 3856, 3924, 3925) and fixes for reported issues.

Peer review result: Overview

The audit team reported a total of 2 findings, of which (with decreasing impact)

- 0 were critical,
- 0 were high,
- 0 were medium,
- 0 were low, and
- 2 were informational.

Peer review result: Detailed findings

Informational: `Redelegate` withdraws 1-2x rent to the reserve

Description

The `Redelegate` instruction in the Stake Pool program - not to be confused with the Stake program's `Redelegate` instruction - allows redelegating stake from one validator to another within the same epoch, without having to wait for the decreased stake to fully deactivate before increasing the stake on the destination validator.

During `Redelegate`, part of the stake decreased from the source validator is used as rent to fund two transient accounts: The source transient account and the ephemeral account, which will be split or merged into the destination transient account.

The source transient account's rent will be withdrawn to the reserve after deactivation, which effectively reduces the amount of staked funds by 1x the rent exemption minimum.

The ephemeral account is split or merged into the destination transient account, depending on whether it already exists. If the destination transient account does *not* exist, then the ephemeral account's rent will be reused as the destination transient account's rent. This rent, however, will not be added to the destination validator stake but will instead be withdrawn to the reserve during an update after the eventual merge. If the destination transient account *does* exist, then the ephemeral account's rent will be added to the validator's stake.

In total, 1-2 times the rent exemption minimum will be deactivated and withdrawn to the reserve when using `Redelegation`.

This does not cause lasting issues, as the staker can compensate with an additional `Increase` instruction afterwards. However, it might be confusing that a validator-to-validator redelegation changes the amount of staked funds, and should at least be documented.

Resolution

This unexpected behaviour was acknowledged and documented in PR #3986. The relevant commit hash is 83cc57517b82e011fede442be017fffe54d92a44.

We think the proposed change is sufficient to address this issue.

Informational: `Redelegate` implicitly checks the destination transient account's state / Decreased funds can be added back in unexpectedly

Description

When merging an ephemeral account into an existing transient account, as is required for `Redelegate`, `IncreaseAdditionalValidatorStake`, and `DecreaseAdditionalValidatorStake`, the ephemeral account needs to have the correct state for the merge to succeed. For `Redelegate` and `IncreaseAdditionalValidatorStake`, the destination transient account is expected to be `Activating`. For `DecreaseAdditionalValidatorStake`, it is expected to be `Deactivating`.

This check is not performed explicitly. Instead, the Stake Pool program relies on the `Merge` instruction of the Stake program to check the destination transient account's state. This is documented for both `IncreaseAdditionalValidatorStake` and `DecreaseAdditionalValidatorStake` but not for `Redelegate`. We recommended adding the relevant documentation for `Redelegate` to avoid introducing a bug here in the future.

During the review of the solution proposed by the SPL Stake Program developers, we were informed by the developers of unexpected behaviour related to this finding that was reported by a 3rd-party auditor. We were asked to confirm this additional finding, which we did, and reviewed the proposed solution. This finding is described in the following.

When a validator has just been added in the current epoch, its canonical stake account will be `Activating` and its transient account will not exist. By using the new `IncreaseAdditionalValidatorStake` instruction, a transient account can be created that is also `Activating`.

When the staker now tries to decrease stake on the validator's canonical stake account via `DecreaseAdditionalValidatorStake`, the instruction will succeed, but the funds will not be stopped from activating. Instead, the funds will be split off from the canonical stake account into an ephemeral stake account and deactivated, at which point the ephemeral account is `Inactive`. Afterwards, the program will attempt to merge the ephemeral account into the destination transient account. This will succeed and cause the funds in the ephemeral account to be added to the `Activating` transient account, essentially reactivating the just-deactivated funds.

Resolution

Our initial finding was addressed in PR #3986 by adding the suggested documentation, which we accepted.

The proposed solution for the finding reported by the 3rd-party auditor is included in PR #3987. This change introduced explicit checks on the destination transient account's state for all three

above-mentioned instructions. For `DecreaseAdditionalValidatorStake`, the transient account is required to be `Deactivating` or `Inactive`. We think this is a sufficient check. For `IncreaseAdditionalValidatorStake` and `Redelegate`, the transient account is required to be `Activating` or `Active`. We have noted that it is not possible for a merge to succeed if the transient account is `Active`, as merging an `Activating` ephemeral account into an `Active` transient account is invalid and will be rejected by the Stake program. Furthermore, a transient account is only ever `Active` if the stake pool is out-of-date, which prevents any of the three instructions to run in the first place.

PR #4002 addresses our concerns by making both checks more strict, narrowing the allowed states of the destination transient account to `Deactivating` and `Activating` respectively. We think this change is sufficient.

Neodyme AG

Dirnismaning 55
Halle 13
85748 Garching
E-Mail: contact@neodyme.io

https://neodyme.io