Peer Review – SPL Stake Pool Updates

Neodyme AG

2022-12-10

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
Update program to work with minimum delegation	https://github.com/solana-labs/solana-program-library/pull/3127
Add / remove validators from the reserve	https://github.com/solana-labs/solana-program-library/pull/3714
instruction to add metadata for pool token	https://github.com/solana-labs/solana-program-library/pull/3335
Use stake program minimum delegation	https://github.com/solana-labs/solana-program-library/pull/3547

All reported issues have been fixed as of December 6, 2022
(commit hash `fd92ccf9e9308508b719d6e5f36474f57023b0b2`).
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 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 3127, 3714, 3335, 3547) and fixes for reported issues.

Peer review result: Overview

The audit team reported a total of 7 findings, of which (with decreasing impact)

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

Peer review result: Detailed findings

Critical: `AddValidatorToPool` instruction allows reclaiming reserve by staker (PR 3714)

Description

In PR 3714, the `AddValidatorToPool` now creates the new validator stake account by splitting from the reserve. But there is no check that the reserve has more than zero lamports after this happens. The staker of the pool could thus obtain control of the reserve account:

1. ensure that the reserve contains exactly the amount of lamports required to add a new validator (by increasing/decreasing the stake of some active validators from the reserve)
2. in the same transaction, do both of the following:
    1. add a new validator, which splits all lamports from the reserve –> reserve is an unitialized stake account after this instruction
    2. initialize a new stake account under the control of the staker at the reserve address

Impacts property: Safety, since the staker can now manipulate the lamports balance of the reserve and therefore also the pool token price

Resolution

Fixed by making sure that after splitting the stake for a new validator, the reserve still has more than the minimium reserve lamports.

Implemented in commit db6293a4aefe3feccfbda8a04d3aef5cab22c28a, part of PR 3714.

High: Recovery of funds by removing validators not always possible (PR 3714)

Description

Users can withdraw staked lamports from the reserve or a user-chosen validator via the `WithdrawStake` instruction. When neither the reserve nor any of the validators have active stake beyond the required minimum amount, i.e., rent-exemption plus minimum delegation, the user is allowed to completely unstake and remove a validator to recover the remaining funds.

In order for a user to completely remove a validator, they have to

1. first, bring the validator down to the minimum amount of lamports and
2. then, in a second instruction, withdraw the rest.

However, this requires the user to withdraw an *exact* amount of lamports. Note that the user specifies the amount of *tokens* and not *lamports* when withdrawing. Depending on the value of the pool tokens, exactly matching the amount of lamports is not always possible, making the user unable to withdraw. A malicious manager can exploit this to prevent users from withdrawing from the pool at all.

As an example, assume that

- one token is worth two lamports,
- the reserve is empty, i.e. it is at the rent-exemption minimum,
- the minimum amount of lamports for a validator's stake account is 1,000,000 lamports,
- all validators have 1,000,001 lamports in their stake accounts.

At this point, a user is unable to withdraw. None of the validators are at the minimum amount of lamports, so removing them is impossible. However, to reduce them to the minimum amount, a user would have to withdraw one lamport, which is also impossible because even a single token is worth two lamports.

Impacts property: Safety, because users are unable to withdraw funds at all.

Resolution

Fixed by adding a small tolerance when withdrawing from validators: A validator can be completely removed if it either has the minimum amount of lamports or has less than one token worth of additional lamports beyond the minimum.

Implemented in https://github.com/solana-labs/solana-program-library/pull/3839.

High: SPL token 2022 introduces new manager fee account hostage options (PR 3714)

Description

Withdrawing stake from the pool requires a transfer of fees to the manager. If this transfer fails, the withdraw also fails. The pool needs to ensure that it is impossible for the transfer to fail, or users may be unable to withdraw their stake.

In the previous version of SPL token, the only way for the transfer to fail was if the destination account does not exist or has the wrong mint. However, PR 3714 adds support for SPL token 2022 as the fee account, which is much more complex and supports various extensions with different failure modes.

Impacts property: Safety, because users may be unable to withdraw.

Resolution

Fixed by adding a whitelist for extensions on both the mint of the pool token and the manager fee token account. If the manager fee account has any extension not in the whitelist, the fee transfer is skipped.

Implemented in commit 9c8a2307dea9107e8ed5e3877cb5818b1520c171, part of PR 3714.

Medium: Token value is rounded up to next lamport on withdraw

Description

When withdrawing tokens from a pool the value in lamports of the tokens is calculated as `ceil( tokens * lamports_per_token)`. Because this rounds up to the next whole lamports amount, the value can be greater than the amount of lamports that were deposited to obtain the tokens.

As an example, assume the stake pool is in a state where each token is worth 1.1 lamports. Consider the following actions:

- deposit 3 lamports -> get `floor(3 / 1.1)` = 2 pool tokens
- withdraw 1 pool token -> get `ceil(1.1)` = 2 lamports
- withdraw 1 pool token again -> get `ceil(1.1)` = 2 lamports

So in total, 4 lamports received for 3 lamports deposited.

The impact of this issue is limited because of transaction costs on Solana and the low amount that can be stolen per transaction. With each withdraw, the gain is at most 1 lamport. However, a single transaction costs at least 5000 lamports. By including the transaction in a block produced by an attacker-controlled validator, 2500 lamports of the fee can be recovered. To exploit this profitably it would be necessary to perform more than 2500 withdraws in a single transaction. This however is much more than the compute limit currently allows.

Impacts property: Safety, because stealing from the pool lowers the lamports per token so users may not be able to receive the deposited funds back in full.

Resolution

Fixed by truncating on withdraw instead of rounding up.

Implemented in https://github.com/solana-labs/solana-program-library/pull/3804.

Low: Force-destaked validator accounts reclaimable

Description

The `UpdateValidatorListBalance` instruction checks whether any validator stake accounts are in the `Initialized` state. This could happen when validators are force-destaked. If that is the case, the stake accounts are merged back into the reserve, putting them into the `Uninitialized` state.

The now-uninitialized stake account can be reclaimed with a second instruction within the same transaction. Since `UpdateValidatorListBalance` can be run by anyone, this means that anyone can reclaim that validator's stake account. After that, the validator can no longer be added to the pool, because the contract will try to re-use and initialize the validator's stake account when adding the validator, which will fail.

This finding does not allow an attacker to steal funds, even though the attacker controls one of the stake accounts, as the contract will never use that account if adding the validator fails. However, this attack is persistent, and the affected validator can not be added to the affected stake pool until the contract is upgraded.

Impact: An attacker can permanently prevent a force-destaked validator from being added back to the stake pool.

Resolution

Fixed by requiring the staker to specify a seed suffix when adding a validator. This allows the staker to simply choose another stake account if the previous one was reclaimed.

Implemented in https://github.com/solana-labs/solana-program-library/pull/3714.

Informational: No test that `CleanupRemovedValidators` does not run out of compute budget if removing all validators from max size pool

Description

The `CleanupRemovedValidators` instruction loops over all removed validators. Since this loop is only bounded by the number of removed validators (which cannot be decreased except by calling this instruction), there is the possibility of running out of compute budget. If that happens, there would be no way to ever cleanup removed validators again.

To ensure that this never happens, there should be a test for the worst-case compute budget usage of this function.

Resolution

Fixed by adding a test that removing all validators from a max-size pool is still within the allowed compute budget.

Implemented in https://github.com/solana-labs/solana-program-library/pull/3806.

Informational: `DecreaseValidatorStake` instruction does not check that transient account meets minimum delegation requirement

Description

When the staker decreases a validator's stake via the `DecreaseValidatorStake` instruction, the to-be-unstaked lamports are split off into a transient stake account. This transient stake account is then deactivated, which will complete at the next epoch boundary. Until then, however, the transient stake account must meet the stake program's minimum delegation amount.

`DecreaseValidatorStake` does not check that amount split off into the transient stake account is enough to meet the minimum delegation amount. If it does not, the stake program will cause the transaction to fail.

This finding has no impact on security, and we thus classify it as informational. The staker is able to circumvent this issue by either decreasing the stake by a larger amount of, if desired, removing the validator altogether.

Resolution

Fixed by also checking for the minimum delegation in `DecreaseValidatorStake`.

Implemented in https://github.com/solana-labs/solana-program-library/pull/3805.

Neodyme AG

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

https://neodyme.io