Peer Review – SPL Stake Pool

Neodyme AG

2021-10-16

Contents

Introduction

As part of the Solana peer review process, Neodyme was engaged to do a detailed security audit of the SPL stake pool program. The program is still under active development and can change in the future. We reviewed the state of the program on June 8th. (commit hash 0a85a9a533795b6338ea144e433893c6c0056210).

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 all code parts of the SPL stake pool program.

Peer review result: Overview

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

- 1 were critical,
- 1 were high,
- 5 were medium,
- 0 were low, and
- 4 were informational.

Peer review result: Detailed findings

Critical: Stealing stake by using uninitialized transient stake accounts

Description

The `UpdateValidatorListBalance` instruction leaves transient stake accounts in an uninitialized state after merging. Any user can re-initialize this stake account in the same transaction, so a transient stake account with user-controlled authorized keys can be created. Passing this "fake" transient stake account to the `UpdateValidatorListBalance` instruction again, causes the `transient_stake_lamports` of the validator to increase even though the stake is not controlled by the pool. Therefore, existing pool tokens can then be withdrawn yielding more stake per token than what is actually in the pool, allowing to steal stake.

Violates property: Safety, as stake can be stolen

Resolution

The solana team acknowledged the finding and added checks to prevent the hijacking of transient accounts inside the `UpdateValidatorListBalance` instruction's merging process. They also added a way for the pool to pass in a seed when creating the transient stake account. This allows to pass in a new seed when a transient account gets hijacked.

High: Instant stake activation

Description

Any user can deposit a stake account with extra, non-staked lamports into the pool receiving pool tokens for all lamports. These pool tokens can be redeemed for a fully staked stake account.

Violates property: Fairness, as a malicious user could deposit a large amount of unstaked, borrowed SOL before the epoch boundary and then reedem their tokens after the boundary and repay the borrow, receiving more rewards than they have staked SOL

Resolution

The solana team acknowledged the finding and have rewritten the calculation for the credited pool tokens inside the `Deposit` instruction.

Medium: Locking user funds by changing the manager to an invalid SPL token account

Description

The `SetManager` instruction allows setting a `manager_fee_account` that is not a valid SPL token account. This manager prevents withdraws, holding user funds hostage.

Violates property: Safety, because withdraws are prevented

The manager fee account can only be set by the manager, thus exploiting this requires a malicious manager.

Resolution

The solana team acknowledged the finding and added a check to prevent changing the `manager_fee_account` to an account not owned by SPL.

Medium: Locking user funds by closing the manager fee account

Description

Closing the `manager_fee_account` results in an error each time fees should be paid. This prevents anyone from withdrawing funds.

Violates property: Safety, because withdraws are prevented

The manager fee account can only be closed by the manager, thus exploiting this requires a malicious manager.

Resolution

The solana team acknowledged the finding and added the `check_manager_fee_info` function to check for a valid manager fee account. By checking the manager fee account, the fee payment can be skipped resulting in a successful withdrawal.

Medium: Locking user funds by changing the preferred validator to not be part of the pool

Description

The `preferred_withdraw_vote_address` can be set to a validator that is not part of the pool.

1. Removing the validator from the pool (state of the entry is now `ReadyForRemoval`)
2. Changing the preferred address to that validator by calling the `SetPreferredValidator` instruction
3. Calling the `CleanupRemovedValidatorEntries` instruction

After these steps withdraw is no longer possible as long as there is some active stake, as the withdrawal method fails with an error if the preferred validator cannot be found.

Violates property: Safety, because withdraws are prevented

The preferred validator can only be changed by the manager, thus exploiting this requires a malicious manager.

Resolution

The solana team acknowledged the finding and added checks to the `SetPreferredValidator` instruction preventing to set a non-active validator as the preferred one.

Medium: Bypass withdrawal fee increase limit

Description

The `withdrawal_fee` can only be increased each epoch by the value of `MAX_WITHDRAWAL_FEE_INCREASE`. The `UpdateStakePoolBalance` instruction replaces the `withdrawal_fee` by the `next_withdrawal_fee` allowing the `SetWithdrawalFee` instruction to increase the `withdrawal_fee` again by the value of `MAX_WITHDRAWAL_FEE_INCREASE` in the same epoch. By alternating both instructions it is possible to arbitrarily increase the `withdrawal_fee` in a single epoch.

Violates property: Safety, because withdraws can be prevented by increasing the fee to 100%

The fee can only be changed by the manager, thus exploiting this requires a malicious manager.

Resolution

The solana team acknowledged the finding and added a check to the `UpdateStakePoolBalance` instruction preventing the update of the `stake_pool.fee` and the `stake_pool.withdrawal_fee` in the same epoch as they are set.

Informational: Invalid manager can be set

Description

The `SetManager` instruction does not require the signature from the new manager, while the `Initialize` instruction requires the signature from the manager set. This allows to accidentally brick the pool when nobody has access to a private key for the manager.

Resolution

The solana team acknowledged the finding and now requires the signature of the manager inside the `SetManager` instruction.

Informational: Initializing unusable pool by passing the same account twice

Description

The `stake_pool_account` and `validator_list_account` passed to the `Initialize` instruction can be the same account, resulting in the validator list's account data being overwritten with the stake pool data, leading to an unusable pool (since account type of the validator list is now wrong).

Resolution

The solana team acknowledged the finding and added a check to the `Initialize` instruction to prevent the `stake_pool_account` and the `validator_list_account` being the same account.

Informational: Initializing unusable pool by not passing the SPL token program

Description

If the `token_program_id` passed to the `Initialize` instruction is not the SPL token program, the pool will be unusable since all the instructions require the SPL token program.

Resolution

The solana team acknowledged the finding but told us, they are planning on supporting non solana token programs in the future. That makes this finding not relevant. Nevertheless, they are planning to add a check inside the `Initialize` instruction for now to prevent that.

Informational: Shared pool mint

Description

When using a custom `token_program_id`, it is possible to initialize two stake pools with the same pool mint. This is not possible with SPL token mints, since initialize checks that the authorized key for the mint is a program derived key depending on the stake pool address. But with a custom `token_program_id`, the authorized key of the mint might be changed by the program later in some way. Since pools with custom `token_program_ids` are currently unusable anyway, this is not a security issue, but may become an issue in the future if stake pools were to be extended to support non-SPL `token_program_id`.

Resolution

The solana team acknowledged the finding but told us, that because non solana token programs are currently not supported. That makes this finding not relevant at the moment.

Neodyme AG

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

https://neodyme.io