Peer Review – SPL Single Stake Pool

Neodyme AG

2023-08-08

Contents

Introduction

As part of ongoing effors to secure the ecosystem together with the Solana Foundation, Neodyme reviewed the recent addition of the Single Stake Pool contract to the Solana Program Library. The review started on 20th of July 2023, and was concluded one week later.

The Single Stake Pool Contract allows a user, as the name implies, to stake to a single validator in a pool with other uses. Conceptually this is similar to the default Solana Program Library (SPL) stake pool, but only for one specific validator at a time. As such, the codebase is a lot smaller. It was found to be concise without superfluous functionality, minimizing the risk of bugs.

In this report, we'll briefly give a technical overview of the functionality and describe one informational finding. There were no other findings.

Scope

In scope is the new on-chain contract single-pool, located in the Solana Program Library on revision `735d7292e35d35101750a4452d2647bdbf848e8b`.

- https://github.com/solana-labs/solana-program-library/tree/master/stake-pool/single-pool

Contract Functionality

Exactly one single-stake-pool can be created for each validator at a canonical address. After creation, everyone can deposit and withdraw active stake accounts from the pool, receiving pool tokens in return. These tokens are liquid and freely tradable. At any time, a user can return these tokens to the pool in exchange for an active stake account with proportional value.

The whole contract is largely permissionless, and there is no admin functionality, with the exception of naming the pool token. The contract is intentionally kept minimal and only has five instructions:

- **Initialize**, which creates a new pool for a validator that does not yet have one at a canonical address. This is permissionless so that anyone can create the canonical pool for any validator.
- **DepositStake**, which users use to hand over an active stake account to the pool. The account gets merged with the existing pool stake, and new pool tokens are minted to the user. The rent of the deposited account gets transferred back to the user.
- **WithdrawStake**, which a user utilizes to burn his pool tokens in return for an active stake account from the pool. The account gets split from the existing pool stake, and authority is transferred to the user.
- **CreateTokenMetadata** is a permissionless utility function that allows anyone to attach a Symbol and Name to the pool token mint, making it easier to track in wallets for users.
- **UpdateTokenMetadata** is the only permissioned instruction. It allows the owner of a vote account to change the name, symbol and uri of the pool token. This does not affect any funds and is purely for off-chain usage.

All of these instructions do thorough account checks. Every PDA uses different seeds, and there can be no collisions. All accounts are ultimately derived from a validator's vote-key, with unique derivation paths for all accounts and authorities.

There is an external dependency on the metaplex metadata program, required for naming tokens. The metadata is only touched during the two optional metadata instructions, so an upgrade of the metadata program cannot break the pool. During metadata creation, it is required for the mint authority to sign. However, the passed accounts are too restrictive to allow attacks, even if the metadata program is taken over maliciously. Specifically, it is impossible to pass the token-program into any of the available accounts, and without the token program, no changes to any token/mint accounts can be made, so the mint signature cannot be abused here.

There currently is no way to do instant SOL deposits or withdrawals, always requiring an active stake account. While this is subject to change on future stake program development, it currently succinctly guarantees that there can be no instant unstake issues or issues with depositing and withdrawing around the epoch-boundary. This does create a limitation that all deposits and withdrawals must happen in sizes larger than the minimum delegation, though, which is currently set at 1 SOL.

Findings

Informational I01 - In times of global warmup-rate limit, newly created pools cannot be deposited into until all stake is activated

Solana has a global stake warmup rate limit, as described in the Solana Documentation. It is currently set to 25% of the active stake. This means that if more than 25% of stake is de/activated in a single epoch, it will take multiple epochs to take effect.

The stake program disallows the merging of so-called transient stake accounts, that is, accounts that are partially de/activated. As the single-stake-pool program directly merges stake accounts from users into the pool stake account, deposits will be disabled when the pool stake is transient.

This can only ever happen when a pool is newly created and only for the rate-limited epoch. Once the pool stake is activated once, it will forever stay activated.

This is an inherent limitation of the stake-program, so it affects all stake-pools that directly merge stake. Any workarounds likely wouldn't be worth it to implement, as warmup limits are rare, and users can simply deposit one epoch later. Withdraws are unaffected, and users will get a partially activated share of the pool account, the same as if they had kept their stake account without the pool.

Methodology

Neodyme's audit team performed a comprehensive examination of the new SPL single 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

Neodyme AG

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

https://neodyme.io