Peer Review – Marinade on Solana

Neodyme AG

2021-09-23

Contents

Introduction

As part of the Solana peer review process, Neodyme was engaged to do a detailed security audit of the marinade on-chain program. A limited review was performed on June 17th. A more thorough one was done from June 30th to July 12th (commit hash 0176157cda888b0c0b92b22f3178a580032ff703).

The audit revealed no major vulnerabilities, but a few low-priority findings were reported and subsequently fixed by the marinade team. This report describes these findings in detail.

Project summary

Marinade provides a liquid staking solution for Solana.

Users can deposit SOL to the marinade contract, which is then staked to a permissioned set of validators according to an administrator-specified distribution. The user receives mSOL, a token representing their share of the staked value, which can be freely traded while the staked SOL is accruing staking rewards. To end staking, the user can either initiate unstaking of their SOL and wait for the epoch boundary before withdrawing it from the contract reserve, or they can use the in-built liquidity pool of marinade to immediately exchange their mSOL for SOL. The latter option incurs a variable fee.

Methodology

Neodyme's audit team performed a comprehensive examination of the marinade contract. The audit team, which consists of security engineers with extensive experience in Solana smart contract security, reviewed and tested the code of the on-chain contract, 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 on-chain marinade program. Since the code repository is currently kept private, we omit the list of audited files and the audited commit hash.

Peer review result: Overview

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

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

Peer review result: Detailed findings

Finding descriptions in this report is kept broad and not reference specific parts of the code due to the code of the marinade contract being closed source.

Low: Arbitrary unstake fee through integer underflow or cast truncation

Description

The liquidation pool instruction attempts to cap unstake fees below a specific value. It does so by checking that both, the minimum and maximum fee, are under that value. However, there is no check that the minimum fee is smaller than the maximum fee. This can lead to an integer underflow in the calculation of the linearly interpolated unstake fee. By setting both values to specific values, a malicious pool creator can set the unstake fees arbitrarily.

Resolution

The marinade team stated that they used `overflow-checks=true` in their `Cargo.toml`, which prevents this vulnerability. This was verified by the audit team. The marinade team additionally added a check that the minimum fee is below the maximum fee to be safe.

The audit team discovered a similar vulnerability regarding this missing check, which off-loaded the error from an arithmetic underflow operation to an unchecked cast truncation. These truncations are not checked by the rust compiler, even when `overflow-checks` are enabled. They simply truncate the value to the appropriate size. This would have allowed a malicious pool creator to still set the unstake fee arbitrarily. However, the check added by the marinade team prevents this.

Informational: Inconsistent maximum reward fee

Description

During initialization of the marinade contract, reward fees are capped to 10000 bps. However, in the `ConfigMarinade` instruction, which may be called at any point after initialization to change the fee structure, reward fees are instead capped to 5000 bps. This is inconsistent. Additionally, both values are surprisingly high.

Resolution

The marinade team acknowledged the inconsistency. They restructured their reward fee checks to always ensure a cap of 10%.

Informational: Anchor usage is inconsistent

Description

The marinade contract uses the anchor framework, which provides utilities to automatically check account properties such as ownership or signer status. However, the usage of anchor in the marinade contract is inconsistent and confusing. In some instructions, anchor is used extensively, while in others, custom checks were written to re-implement some of anchor's functionality. This makes it harder to verify that all relevant account checks were performed.

Resolution

The marinade team acknowledged the finding and said they would check how to resolve this.

Neodyme AG

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

https://neodyme.io