Security Audit – Marinade Validator Bonds

conducted by Neodyme AG

Nico Gründel (Lead Auditor)

Simon Klier (Second Auditor)

February 4^{th} 2025

Table of Contents

1 Introduction

Neodyme audited changes to **Marinades** on-chain Validator Bond program during December of 2024 and January of 2025. The scope of this audit was focused on technical security, with further considerations about operational security. The auditors found that Marinade's Validator Bond program comprised a clean design and above-standard code quality, relying on the industry-standard Anchor framework. **No issues** were found.

Summary of Findings

During the audit, **no security-relevant** findings were identified.

2 | Scope

The contract audit's scope is focused on the **Implementation** security of the changes to the contract's source code.

Neodyme considers the source code, located at https://github.com/marinade-finance/validator-bonds, in scope for this audit. Third-party dependencies are not in scope.

The relevant source code revisions are:

- `7e6d35e8337174bfe6fcf2691914ac65427f6095` • Start of the audit
- `4a5b009fd5a50774a99225939393c1f4e2e71a1b` • Last reviewed revision

The relevant pull requests are:

Name	URL
fund settlement with non-delegated lamports	https://github.com/marinade-finance/validator-bonds/pull/77
using bitmap data structure instead of PDA	https://github.com/marinade-finance/validator-bonds/pull/73
fund makes possible when activating	https://github.com/marinade-finance/validator-bonds/pull/74
fix for claim settlement on exact match	https://github.com/marinade-finance/validator-bonds/pull/70
merge stake fixing correct signature for merging	https://github.com/marinade-finance/validator-bonds/pull/69

3 | Overview

This section briefly outlines the reviewed changes to the Validator Bonds contract.

Pull Request	Description
fund settlement with non-delegated lamports	Allow funding Settlement accounts with a stake accounts with a larger than necessary balance.
using bitmap data structure instead of PDA	Use bitmap to track claimed settlements instead of individual PDAs.
fund makes possible when activating	Allow funding of inactive stake accounts.
fix for claim settlement on exact match	Do not charge the rent of a stake account when claiming the entire account.
merge stake fixing correct signature for merging	Fixes an issue which prevented merging stake accounts owned by the settlement staker authority.

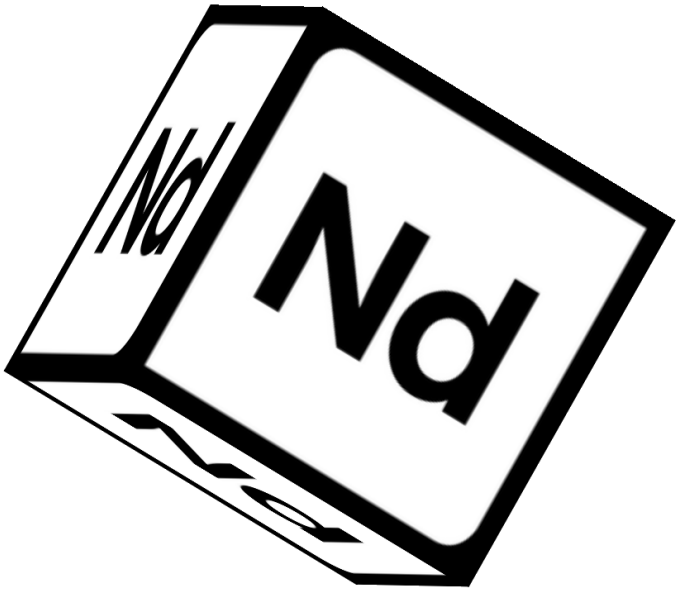
A | About Neodyme

Security is difficult.

To understand and break complex things, you need a certain type of people. People who thrive in complexity, who love to play around with code, and who don't stop exploring until they fully understand every aspect of it. That's us.

Our team never outsources audits. Having found over 80 High or Critical bugs in Solana's core code itself, we believe that Neodyme hosts the most qualified auditors for Solana programs. We've also found and disclosed critical vulnerabilities in many of Solana's top projects and have responsibly disclosed issues that could have resulted in the theft of over $10B in TVL on the Solana blockchain.

All of our team members have a background in competitive hacking. During such hacking competitions, called CTFs, we competed and collaborated while finding vulnerabilities, breaking encryption, reverse engineering complicated algorithms, and much more. Through the years, many of our team members have won national and international hacking competitions, and keep ranking highly among some of the hardest CTF events worldwide. In 2020, some of our members started experimenting with validators and became active members in the early Solana community. With the prospect of an interesting technical challenge and bug bounties, they quickly encouraged others from our CTF team to look for security issues in Solana. The result was so successful that after reporting several bugs, in 2021, the Solana Foundation contracted us for source code auditing. As a result, Neodyme was born.



B | Methodology

Neodyme prides itself on not being a checklist auditor. We adapt our approach to each audit, investing considerable time into understanding the program upfront and exploring its expected behavior, edge cases, invariants, and ways in which the latter could be violated. We use our uniquely deep knowledge of Solana internals, and our years-long experience in auditing Solana programs to even find bugs that others miss. We often extend our audit to cover off-chain components in order to see how users could be tricked or the contract affected by bugs in those components.

Nonetheless, we also have a list of common vulnerability classes, which we always exhaustively look for. We provide a sample of this list below.

- Rule 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
- Check for unsafe design decisions that might lead to vulnerabilities being introduced in the future
- Check for any other, as-of-yet unknown classes of vulnerabilities arising from the structure of the Solana blockchain
- Ensure that the contract logic correctly implements the project specifications
- Examine the code in detail for contract-specific low-level vulnerabilities
- Rule out denial of service attacks
- Rule out economic attacks
- Check for instructions that allow front-running or sandwiching attacks
- Check for rug pull mechanisms or hidden backdoors

C | Vulnerability Severity Rating

Critical Vulnerabilities that will likely cause loss of funds. An attacker can trigger them with little or no preparation, or they are expected to happen accidentally. Effects are difficult to undo after they are detected.

High Bugs that can be used to set up loss of funds in a more limited capacity, or to render the contract unusable.

Medium Bugs that do not cause direct loss of funds but that may lead to other exploitable mechanisms, or that could be exploited to render the contract partially unusable.

Low Bugs that do not have a significant immediate impact and could be fixed easily after detection.

Info Bugs or inconsistencies that have little to no security impact.

Neodyme AG

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

https://neodyme.io