Security Audit - Marinade Validator Bonds PR Review

conducted by Neodyme AG

Lead Auditor:	Nico Gründel
Second Auditor:	Simon Klier
Supporting Auditor:	Jasper Slusallek
Administrative Lead:	Thomas Lambertz

February 17, 2025

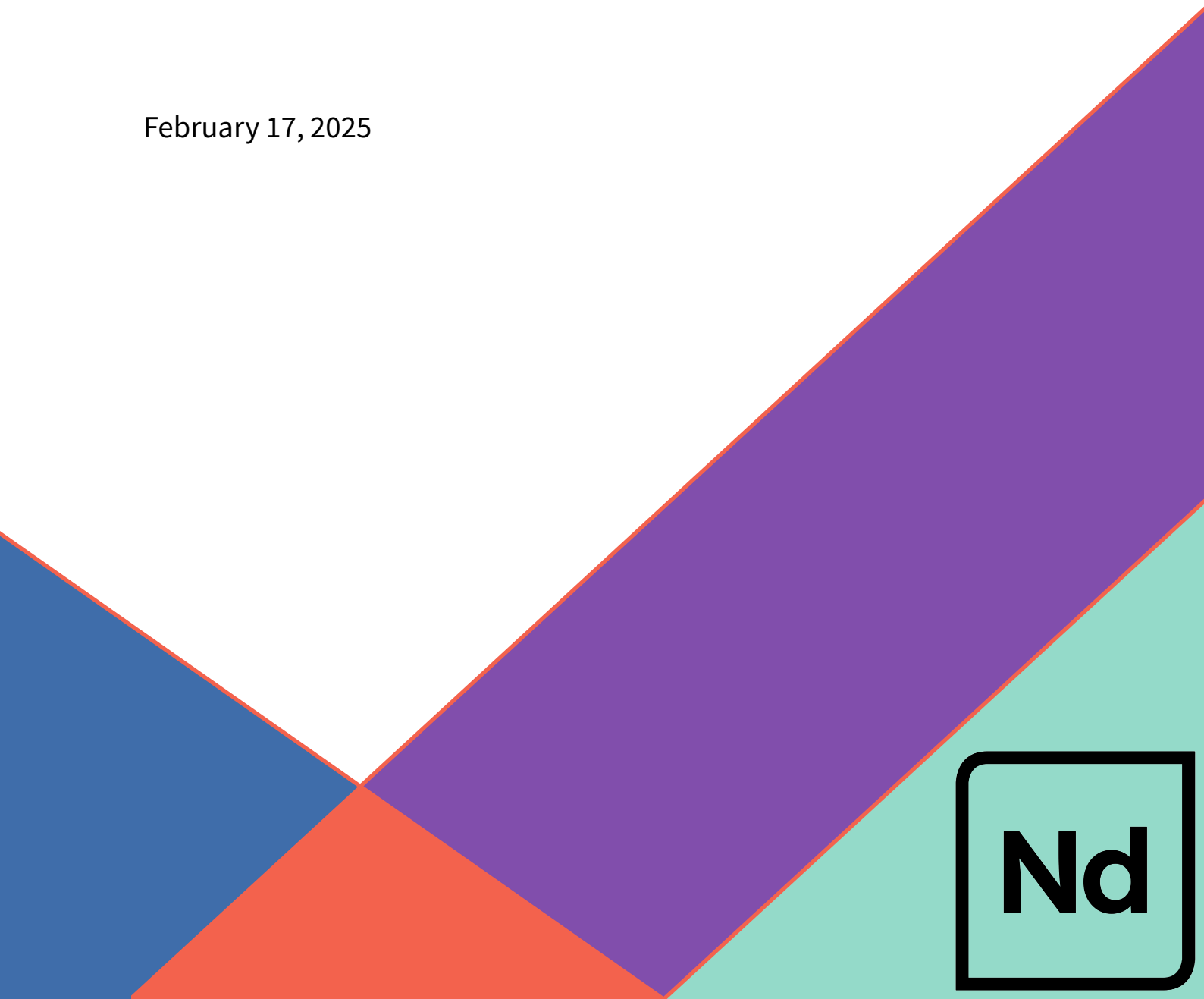

Table of Contents

1 | Executive Summary

Neodyme audited changes to **Marinade's** on-chain Validator Bonds program during December of 2024.

The volume of changes was small.

The auditors found that the Validator Bonds program's clean design persisted with the changes, with only minor new complexity being introduced. The program relies on the industry-standard Anchor framework.

No issues of note were found by the auditors. This is as uncommon result in Neodyme audits, and should be positively acknowledged despite the small volume of changes.

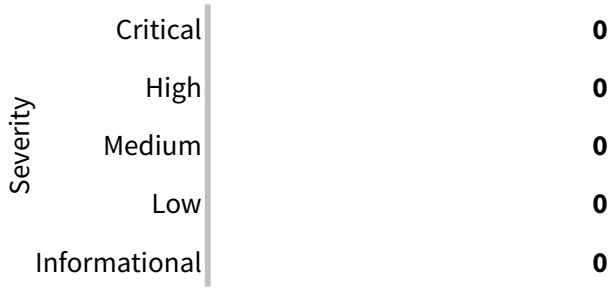

Figure 1: Overview of Findings

For a full list and summary of audited changes, see the Scope section. The scope of this audit included both the implementation security of the changes themselves, as well as any security-relevant effects they may have on the already existing program.

2 | Introduction

In August of 2024, Marinade Finance introduced Delegation Strategy v2, which transitions the Marinade liquid staking system to use the so-called Stake Auction Marketplace (SAM). This approach allocates stake controlled by Marinade such that staking return is maximized. It takes into account both the staking returns themselves as well as bids that validators place in order to receive higher delegations. MNDE-directed votes also control parts of the staking strategy .

The Validator Bonds contract audited here is a component of Marinade's Protected Staking Rewards (PSR) system. It was conceived to safeguard stakers against potential yield losses due to underperformance or intentional changes by the validators. By establishing a bond, validators commit a portion of their own SOL as collateral. This bond is used to ensure that any rewards lost from e.g. downtime or commission changes are compensated. The required bond amount is directly proportional to the stake a validator receives from Marinade. This bond must be established before a validator is eligible to receive stake from Marinade.

The changes that were reviewed in this audit **do not change this core functionality**. Instead, they are small changes that modify or fix technical details of the contract.

Marinade engaged Neodyme to do a full and detailed audit of the changes. In accordance with Marinade's wishes, the auditors focused on **technical security and compatibility with the existing code**, along with further considerations about operational security.

The auditing team consisted of two senior auditors from Neodyme, Nico Gründel and Simon Klier. Supporting partial reviews were performed by Jasper Slusallek. All three of these auditors have a track record of finding critical and other vulnerabilities in Solana programs, and have audited many similar programs. Nico Gründel and Simon Klier were additionally the auditors for the original full audit of the Validator Bonds program in March and April of 2024 and thus have a deep unterstanding of the program's code and its edge cases.

The changes reviewed – as laid out in the section hereafter – are small and kept to the minimum for each of the desired effects.

3 | Scope

The primary objective of this audit was to evaluate the implementation security of changes made to the on-chain Validator Bond contract's source code.

The pull requests relevant to this audit were explicitly specified by the Marinade team and communicated to the auditors in writing. We list the pull requests in the table below for reference.

id	description	link
PR #77	Allow funding Settlement accounts with a stake account with a larger than necessary balance	link to PR
PR #73	Use bitmap to track claimed settlements instead of individual PDAs	link to PR
PR #74	Allow funding of inactive stake accounts	link to PR
PR #70	Do not charge the rent of a stake account when claiming the entire account	link to PR
PR #69	Fixes an issue which prevented merging stake accounts owned by the settlement staker authority.	link to PR

The last reviewed commit was: `4a5b009fd5a50774a99225939393c1f4e2e71a1b`

This assessment was focused on identifying any bugs, unintended security risks, or deviations from best practices that may have been introduced as a result of these changes. Additionally, this audit considered the potential impact of these modifications on the contract's existing logic. Neodyme aimed to prevent that these changes inadvertently introduce security weaknesses or alter the contract's expected behavior.

To achieve this, the auditors examined the relevant changes in detail. The scope of this audit was strictly limited to the specific modifications outlined above, as well as their interactions with the existing contract code. It does not extend to third-party dependencies, external integrations, or components unrelated to the specified changes.

The contract, along with the pull requests relevant to this audit, is publicly available in the following GitHub repository: https://github.com/marinade-finance/validator-bonds

4 | Findings

The audit of the changes revealed **no findings** of relevance for security.

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

We are not checklist auditors.

In fact, we pride ourselves on that. 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 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 here.

Select Common Vulnerabilities

Our most common findings are still specific to Solana itself. Among these are vulnerabilities such as the ones listed below:

- Insufficient validation, such as:
  - Missing ownership checks
  - Missing signer checks
  - Signed invocation of unverified programs
  - Account confusions
  - Missing freeze authority checks
  - Insufficient SPL account verification
  - Dangerous user-controlled bumps
  - Insufficient Anchor account linkage
- Account reinitialization vulnerabilities
- Account creation DoS
- Redeployment with cross-instance confusion
- Missing rent exemption assertion
- Casting truncation
- Arithmetic over- or underflows
- Numerical precision and rounding errors
- Anchor pitfalls, such as accounts not being reloaded
- Non-unique seeds
- Issues arising from CPI recursion
- Log truncation vulnerabilities
- Vulnerabilities specific to integration of Token Extensions, for example unexpected external token hook calls

Apart from such Solana-specific findings, some of the most common vulnerabilities relate to the general logical structure of the contract. Specifically, such findings may be:

- Errors in business logic
- Mismatches between contract logic and project specifications
- General denial-of-service attacks
- Sybil attacks
- Incorrect usage of on-chain randomness
- Contract-specific low-level vulnerabilities, such as incorrect account memory management
- Vulnerability to economic attacks
- Allowing front-running or sandwiching attacks

Miscellaneous other findings are also routinely checked for, among them:

- Unsafe design decisions that might lead to vulnerabilities being introduced in the future
    - Additionally, any findings related to code consistency and cleanliness
- Rug pull mechanisms or hidden backdoors

Often, we also examine the authority structure of a contract, investigating their security as well as the impact on contract operations should they be compromised.

Over the years, we have found hundreds of high and critical severity findings, many of which are highly nontrivial and do not fall into the strict categories above. This is why our approach has always gone way beyond simply checking for common vulnerabilities. We believe that the only way to truly secure a program is a deep and tailored exploration that covers all aspects of a program, from small low-level bugs to complex logical vulnerabilities.

C | Vulnerability Severity Rating

We use the following guideline to classify the severity of vulnerabilities. Note that we assess each vulnerability on an individual basis and may deviate from these guidelines in cases where it is well-founded. In such cases, we always provide an explanation.

Severity	Description
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 exploits 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.
INFORMATIONAL	Bugs or inconsistencies that have little to no security impact but are still noteworthy.

Additionally, we often provide the client with a list of nit-picks, i.e. findings whose severity lies below Informational. In general, these findings are not part of the report.

Neodyme AG

Dirnismaning 55
Halle 13
85748 Garching
Germany

E-Mail: contact@neodyme.io

https://neodyme.io