Pod Network Bridge Update + DepositWaitingList Audit Report

Prepared by Riley Holterhus

February 25, 2026

Contents

1 Introduction

1.1 About Pod Network

Pod Network is a Layer-1 network with an architecture that differs from most traditional blockchains. One notable aspect is that transactions are partially ordered rather than enforced under a strict total order, which allows for faster transaction processing.

1.2 About the Auditor

Riley Holterhus is an independent security researcher who focuses on Solidity smart contracts. Other than conducting independent security reviews, he works as a Lead Security Researcher at Spearbit, and also searches for vulnerabilities in live codebases. Riley can be reached by email at rileyholterhus@gmail.com, by Telegram at @holterhus and on Twitter/X at @rileyholterhus.

1.3 Disclaimer

This report is intended to detail the identified vulnerabilities of the reviewed smart contracts and should not be construed as an endorsement or recommendation of any project, individual or entity. While the authors have made reasonable efforts to detect potential issues, the absence of any undetected vulnerabilities or issues cannot be guaranteed. Additionally, the security of the smart contracts may be affected by future changes or updates. By using the information in this report, you acknowledge that you are doing so at your own risk and that you should exercise your own judgment when implementing any recommendations or making decisions based on the findings. This report has been provided on an “as-is” basis and DOES NOT CONSTITUTE A GUARANTEE OR WARRANTY OF ANY FORM.

2 Audit Overview

2.1 Scope of Work

On February 20, 2026, Riley Holterhus conducted a manual security review of Pod's `Bridge` and `DepositWaitingList` smart contracts. The goal was to identify potential vulnerabilities and logic issues in the codebase.

The audit was performed on the codebase found in the `podnetwork/pod-sdk` GitHub repository. The audit started on commit 552020a. The files in scope for the audit were:

- `protocol/src/Bridge.sol`
- `protocol/src/DepositWaitingList.sol`

The `Bridge` contract was reviewed in a prior audit. Changes since the previous review include renaming "deposit" functions and strings to "claim", refactoring several functions without altering their core behavior, and incorporating `chainid()` into claim hash calculations. The `chainid()` addition reflects the intent to deploy `Bridge` contracts across multiple chains. Notably, the Pod team intends to support each asset on a single chain only - there should never be a case where the same asset can be bridged to/from multiple chains.

The `DepositWaitingList` contract is a wrapper around the Bridge that decouples user deposits from the bridge capacity. Users deposit funds in a single step and a relayer later forwards them to the Bridge when capacity is available. This allows users to initiate deposits even while the `Bridge`'s rate limits are fully utilized.

2.2 Summary of Findings

Each finding from the audit has been assigned a severity level of "Critical", "High", "Medium", "Low" or "Informational". These severities aim to capture the impact and likelihood of each potential issue.

In total, **4 findings** were identified, all being informational. The description and status of the findings are provided in the sections below.

3 Findings

3.1 Informational Findings

3.1.1 `DepositWaitingList` has more restrictive deposit limits

Description: The new `DepositWaitingList` contract's `deposit()` function validates against the bridge's `depositLimit`:

```
(uint256 minAmount, uint256 depositLimit,,,,) = bridge.tokenData(token);
if (amount < minAmount || amount > depositLimit) revert InvalidDepositAmount();
```

This check is more restrictive than the Bridge's own logic. This is because the Bridge accepts amounts larger than `depositLimit` in a single call if a period reset is triggered. Such deposits can use the remaining capacity from the old 24-hour window and allocate the excess against the newly-reset usage.

Recommendation: Consider whether the `DepositWaitingList` should have more restrictive deposit rules than the underlying `Bridge`. It's possible this is an intended design choice.

Pod: Acknowledged.

Auditor: Acknowledged. After discussing with the team, the `DepositWaitingList` being more restrictive is not an issue.

3.1.2 Solidity version has high severity bug

Description: A high-severity Solidity compiler bug affects versions 0.8.28 - 0.8.33 when compiled using the IR pipeline. The project's `foundry.toml` meets both conditions:

```
...
solc = "0.8.28"
...
via_ir = true
...
```

However, this issue does not impact the codebase in practice, as the bug only occurs when deleting transient storage variables, which are not used in this project.

Recommendation: As a precaution, consider upgrading to Solidity 0.8.34, where the issue is fixed. If upgrading is not planned, consider documenting this behavior in case transient storage is introduced in future versions.

Pod: Bumped Solidity version in PR 171.

Auditor: Verified.

3.1.3 `withdraw()` can be used to grief `applyDeposits()` batch transactions

Description: A depositor can front-run a pending `applyDeposits()` transaction that includes their `depositId` by calling `withdraw()` first. This deletes `depositHashes[depositId]`, causing the entire batch to revert on the `DepositNotPending` check, temporarily delaying other depositors in the same batch.

The impact is limited to temporary griefing at the cost of the attacker's own gas. The relayer recovers by resubmitting the batch without the withdrawn deposit.

Recommendation: Consider skipping deposits in batch that no longer exist, instead of reverting the entire batch.

Pod: Addressed in commit 448ea31,

Auditor: Verified.

3.1.4 Deposit hash doesn't commit to `callContract` or `reserveBalance`

Description: The user's deposit hash commits only to `token`, `amount`, `from`, and `to`. The `callContract` and `reserveBalance` arguments passed by the relayer to `applyDeposits()` have no on-chain binding to depositor intent. In practice, the relayer is a trusted role and would be assumed to provide these values how the depositors want. However, to reduce the trust on the relayer, the `callContract` and `reserveBalance` values could also be added to the hashes committed on-chain.

Recommendation: Consider having depositors commit to `callContract` and `reserveBalance` at deposit time by including them in the deposit hash.

Pod: Addressed in commit 5e15464.

Auditor: Verified.