Security Research Labs

Liquidity-Pools Code Review

Security audit of Centrifuge's multi-chain liquidity pool protocol

V1.1, October 3, 2023

Mostafa Sattari mostafa@srlabs.de
Kevin Valerio kevin@srlabs.de
Regina Bíró regina@srlabs.de

Content

1 Context

This report is intended to provide a comprehensive account of the processes, findings, and methodology employed during the security audit conducted on Centrifuge's Liquidity-Pools logic that allows users from various blockchain ecosystems to interact with the Centrifuge parachain's business logic deployed on Polkadot.

2 Scope and methodology

The security assessment was primarily focusing on the logic flow between the Liquidity Pools Solidity smart-contract, the Gateway implementation for Axelar and the Foreign Investments pallet interfacing with Centrifuge's business logic. Typical issues regarding smart contract and Solidity coding practices were excluded from the scope of this audit, as a dedicated Code4rena assessment was performed to cover these [1]. The below list includes various components that were considered part of the review's scope.

Component name	Reference	Identified issues
Ethereum Transaction	https://github.com/centrifuge/centrifuge-chain/tree/main/pallets/ethereum-transaction	No issue was found.
Foreign Investments	https://github.com/centrifuge/centrifuge-chain/tree/main/pallets/foreign-investments	[2]
Liquidity Pools (pallet logic)	https://github.com/centrifuge/centrifuge-chain/tree/main/pallets/liquidity-pools	[3]
Liquidity Pools Gateway	https://github.com/centrifuge/centrifuge-chain/tree/main/pallets/liquidity-pools-gateway	[4]
Liquidity Pools (smart contract)	https://github.com/centrifuge/liquidity-pools	No issue was found.

The primary focus of the security audit mainly involved manual code review. Furthermore, in addition to manual code review, the in-scope components have undergone testing using SRLabs' in-house static and dynamic analysis tooling.

3 Findings details

The following paragraph summarizes the comprehensive findings revealed during the security audit, with the most recent update time for each status being October 3, 2023.

Issue title	0-weight for message processing enables DoS
Tracking	[4]
Severity	Medium
Status	Fixed

In the liquidity-pools-gateway implementation, the *process_msg* extrinsic implements the logic of processing incoming messages and adding them to the inbound queue after performing some sanity checks. While the extrinsic can only be invoked by trusted relayers, the assigned weight of 0 to the extrinsic enables DoS attacks, as this weight does not account for the computational complexity of processing the incoming message.

Even if this issue cannot be exploited by malicious actors directly, the bug can cause a partial denial of service, where Centrifuge's block production halts until the *process_msg* call causing the issue is filtered out or paired with less heavy extrinsics in a block. As a result, the bug could also cause a delay in processing incoming messages from the Axelar bridge.

The issue was fixed by replacing the 0 value with a corresponding weight function [5], which uses a unified weight overestimation for each call type that will be processed. In the future, we recommend benchmarking all message types and implementing a dynamic weight calculation scheme for *process_msg*.

Issue title	Missing benchmarking in Foreign Investments
Tracking	[2]
Severity	Info
Status	Closed

The foreign-investments pallet does not contain any benchmarking and the weights file is empty. Underweighted extrinsics can lead to overweight blocks and cause block production timeouts. As the pallet was under development at the time of reporting this issue, the auditors assigned information-level severity to it.

The Centrifuge team confirmed that as there are no extrinsics implemented for the foreign investments pallet, nor are planned in the future, the issue can be closed. To avoid further confusion, it is planned to completely remove the *weights.rs* file from the codebase.

Issue title	Missing benchmarking in Liquidity Pools
Tracking	[3]
Severity	Info
Status	Open

The liquidity pools pallet does not contain any benchmarking and currently the following extrinsics' assigned weight contains placeholder values:

- *add_currency*
- *allow_investment_currency*
- *schedule_upgrade*
- *cancel_upgrade*
- *update_tranche_token_metadata*

Underweighted extrinsics can lead to overweight blocks and cause block production timeouts. As the pallet was under development at the time of reporting this issue, the auditors assigned information-level severity to it.

While some of the affected extrinsics are permissioned, it is still advisable to properly benchmark and weigh them to avoid overweight blocks.

4 Evolution suggestions

To fortify Centrifuge against both known and unforeseen risks, we strongly advise adhering to the suggested remediation solutions and following the below detailed best practices.

Address remaining security issues. Every reported security issue must be addressed as soon as possible. Even if an open issue seems to have only a minor impact, it is important to consider that malicious actors could potentially exploit it as a component in their attack strategy, potentially leading to more significant repercussions for the overall application.

Address TODOs. As the logic implementing core features of Liquidity Pools is still evolving, there are quite a few TODOs noted in the code comments that need to be addressed in the future. We recommend Centrifuge to address these in a timely manner, as noting missing/immature features might give hints to attackers on how to attempt exploiting the logic in question.

Embrace defensive programming. Use a defensive programming approach to prevent unexpected bugs and mitigate potential security vulnerabilities, for example by using saturating/checked functions when performing arithmetic operations, or perform extensive sanity checks before using raw operators, for example in the Liquidity Pools smart contract.

Establish ecosystem best practice processes. Keep in mind the optimal procedures that are considered best practices within the Substrate ecosystem, such as performing proper benchmarking using the application's runtimes.

Conduct an economic audit. The auditors highly recommend performing an economic audit on the economic parameters of the Centrifuge codebase, specifically examining the economic configuration items and message fees concerning the liquidity-pools feature.

5 Conclusion

The primary goal of this audit was to conduct a comprehensive security assessment of various components, with a primary focus on the liquidity pools implementation. No critical issues were identified during this evaluation. All identified issues were promptly communicated to the Centrifuge team through a dedicated Slack channel and have been officially documented and disclosed in our shared private GitHub repository.

6 Bibliography

[1] [Online]. Available: https://github.com/code-423n4/2023-09-centrifuge/blob/main/bot-report.md.

[2] [Online]. Available: https://github.com/centrifuge/centrifuge-chain-internal/issues/50.

[3] [Online]. Available: https://github.com/centrifuge/centrifuge-chain-internal/issues/51.

[4] [Online]. Available: https://github.com/centrifuge/centrifuge-chain-internal/issues/49.

[5] [Online]. Available: https://github.com/centrifuge/centrifuge-chain/pull/1562.

[6] [Online]. Available: https://github.com/centrifuge/centrifuge-chain/blob/56fe24a5b6b190e50b9e60dccc874cae74c94952/pallets/liquidity-pools-gateway/src/weights.rs#L85.