certora

Security Assessment
Final Report

Balancer

Gyroscope Pools

December 2024

Prepared for Balancer

Table of contents

certora Balancer

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
Balancer V3	balancer-v3-monorepo	2d6ae6a	EVM

Project Overview

This document describes the verification of Balancer V3 Gyroscope pools using manual code review. The work was undertaken from December 12th 2024 to December 24th 2024.

The following files are considered in scope for this review:

pkg/pool-gyro/contracts/Gyro2CLPPool.sol
pkg/pool-gyro/contracts/Gyro2CLPPoolFactory.sol
pkg/pool-gyro/contracts/GyroECLPPool.sol
pkg/pool-gyro/contracts/GyroECLPPoolFactory.sol
pkg/pool-gyro/contracts/lib/Gyro2CLPMath.sol
pkg/pool-gyro/contracts/lib/GyroECLPMath.sol
pkg/pool-gyro/contracts/lib/GyroPoolMath.sol
pkg/pool-gyro/contracts/lib/SignedFixedPoint.sol

The team performed a manual audit of all the Solidity contracts**.** During manual audit, the Certora team discovered issues in the Solidity contracts code, as listed in the following.

Protocol Overview

Balancer v3 is the successor of v2 and is a decentralized automated market maker (AMM) protocol built on Ethereum with a clear focus on fungible and yield-bearing liquidity.
Gyroscope Concentrated Liquidity pools (CLP) are a class of AMMs that price the exchange of assets within a defined range. CLPs are currently built on Balancer V2 and include Quadratic curve pools and Elliptic curve pools.

In this updated version of the pools, the pool contracts have been adapted to be compatible with Balancer V3.
Compared to the V2 version, all token scaling logic has been removed from the pools and is now handled by the vault. Balancer V3 also allows unbalanced liquidity operations, which was not available in V2.

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	3	3	3
Informational	1		
Total	4	3	3

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

Detailed Findings

ID	Title	Severity	Status
L-01	Insufficient checks in validateDerivedParamsLimits	Low	Fixed in 080bdd7
L-02	Inconsistent rounding	Low	Fixed in e292dd4
L-03	Pool registered twice with factory on pool creation	Low	Fixed in 808f6e3
I-01	SignedFixedPoint not optimized for Solidity 0.8.x	Informational	

Low Severity Issues

L-01 Insufficient checks in validateDerivedParamsLimits		
Severity: **Low**	Impact: **Medium**	Likelihood: **Low**
Files: GyroECLPMath.sol	Status: Fixed in 080bdd7	

Description: For the Elliptic Curve Pools, some parameters are calculated offchain and are supplied to the pool constructor on deployment. Supplying inconsistent or incorrect parameters could lead to the pool behaving incorrectly.
Especially incorrect values of $\tau(\alpha)$ and $\tau(\beta)$ have a big impact on the pool calculations and will allow users to take out more than they have deposited or do very favourable swaps.

Even though the documentation states:
"*Note that, while the constructor performs some basic checks on these parameters, it is the deployer's responsibility to ensure that they are indeed consistent with each other.*",
some constraints could be easily checked by the contract.

Recommendations: It is recommended to add the following checks to minimize risk of accidental incorrect parameters:

```
Unset
τ(α)y > 0
τ(β)y > 0
τ(β)x > τ(α)x
```

Customer's response: Confirmed and fixed in commit 080bdd7

L-02 Inconsistent rounding		
Severity: **Low**	Impact: **Low**	Likelihood: **Medium**
Files: Gyro2CLPPool.sol	Status: Fixed in e292dd4	

Description: In `calculateVirtualParameter0`, `calculateVirtualParameter1`, `calculateQuadratic`, `computeBalance` there are rounding issues.

For example In `calculateQuadratic` function:

`denominator = a.mulUp(2 * FixedPoint.ONE)` should round down,

`addTerm = (mc.mulDown(4 * FixedPoint.ONE)).mulDown(a)` should round up.

All calculations should be rounded to the right direction, it does not happen in all of these functions.

Recommendations: correct rounding direction to the right side.

Customer's response: Confirmed and fixed in commit e292dd4

L-03 Pool registered twice with factory on pool creation		
Severity: **Low**	Impact: **Low**	Likelihood: **likely**
Files: Gyro2CLPPoolFactory.sol GyroECLPPoolFactory.sol	Status: Fixed in 808f6e3	

Description: When creating a pool via the `Gyro2CLPPoolFactory` or `GyroECLPPoolFactory`, The `create()` function uses `BasePoolFactory._create()` to create the pool, which includes a call to `_registerPoolWithFactory(pool)`;
At the end of create() an additional `_registerPoolWithFactory(pool)` is done.
This causes each new pool to be added to the `_pools` array twice.

Recommendations: Remove the redundant `_registerPoolWithFactory(pool)` call.

Customer's response: Confirmed and fixed in commit 808f6e3

Informational Severity Issues

I-01. SignedFixedPoint not optimized for Solidity 0.8.x

Description: The original version of this library was written for Solidity 0.7.x
For most functions in the SignedFixedPoint library there is a normal implementation that has specific overflow checks, and an unchecked function for gas optimisation, where no overflow checks are performed.
It is now on Solidity version 0.8.24 so overflow checks for math operations are handled by the solidity compiler. So the normal implementation now has unneeded double checks on overflow and the unchecked versions are still checked.
Because of this the functions consume more gas than needed.

Recommendation: Optimize the code for Solidity 0.8.x.

Customer's response: Balancer is aware that the code has not been optimized for new Solidity versions. It was decided to make as little change as possible to the library to minimize the risk of introducing any new bugs.

Disclaimer

Even though we hope this information is helpful, we provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.