certora

Security Assessment
Final Report

Balancer

Stablepool Surge hooks

January 2025

Prepared for Balancer

Table of contents

Project Summary **3**
Project Scope 3
Project Overview 3
Protocol Overview 3
Findings Summary 4
Severity Matrix 4
Detailed Findings **5**
Low Severity Issues 6
L-01 Roundtrip add/remove liquidity possible to avoid paying the surge fee 6
L-02 Fees are not included in the threshold or fee calculations 7
L-03 Incorrect maxSurgeFeePercentage setting can cause underflow 7
Informational Severity Issues 8
I-01. Users can front-run swaps and force surge pricing for other users 9
Disclaimer **10**
About Certora **10**

certora Balancer

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
Balancer V3	balancer-v3-monorepo	68cc540	EVM

Project Overview

This document describes the verification of Balancer V3 Stablepool Surge hook manual code review. The work was undertaken from January 23rd 2025 to January 30th 2025.

The following files are considered in scope for this review:

pkg/pool-hooks/contracts/StableSurgeHook.sol
pkg/pool-hooks/contracts/StableSurgePoolFactory.sol
pkg/pool-hooks/contracts/utils/StableSurgeMedianMath.sol

During the course of the audit a development branch *stable-surge-disable-unbalanced-liquidity* was created to add code to handle unbalanced liquidity operations. Part of that new code was reviewed and discussed with the Balancer team, but because this code is still in the development phase, it is considered out of scope for this audit.

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 pools support hooks to validate or modify pool transaction data.
For StablePool swap operations, a static swap fee is charged, which is a percentage of the swapped amount. The Stable Surge hook is intended to charge an additional fee to users when they make a swap that imbalances a StablePool above a specified threshold. This dynamic swap fee is calculated based on how much the swap increases the imbalance.

Swaps that reduce the imbalance or swaps where the imbalance remains below a threshold are not charged any additional fees.

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	2
Informational	1	1	0
Total	4	4	2

Severity Matrix

Impact					
	Critical	Medium	Medium/High	High	Critical
	High	Low/Medium	Medium	Medium/High	High
	Medium	Low	Low/Medium	Medium	Medium/High
	Low	Informational	Low	Low/Medium	Medium
		Rare	Unlikely	Likely	Very Likely
			Likelihood		

Detailed Findings

ID	Title	Severity	Status
L-01	Roundtrip add/remove liquidity possible to avoid paying the surge fee	Low	Fixed in 4680d59
L-02	Fees are not included in the threshold or fee calculations	Low	Acknowledged, Will not be fixed.
L-03	Incorrect maxSurgeFeePercentage setting can cause underflow	Low	Fixed in 767a6a1

Low Severity Issues

L-01 Roundtrip add/remove liquidity possible to avoid paying the surge fee		
Severity: **Low**	Impact: **Low**	Likelihood: L**ikely**
Files: StableSurgePoolFactory.sol	Status: Fixed in 4680d59	

Description: The StableSurgePoolFactory create function has a parameter disableUnbalancedLiquidity which makes it possible to create a pool which supports unbalanced liquidity operations.

When a user wants to do a swap that imbalances the pool above the threshold, he will be charged an increased dynamic swap fee.

With unbalanced liquidity enabled, the same swap can also be done via a AddLiquidity proportional and a RemoveLiquidity single token. This makes it possible for the user to do the swap while bypassing the obligation to pay the extra fees.

Recommendations: Disable all unbalanced liquidity operations for all pools that use the SurgeHook or add hooks on the liquidity operations that check if liquidity operations imbalance the pool.

Customer's response: Balancer is working on extending the SurgeHook contract with additional hooks for Add and Remove liquidity. These hooks will block unbalanced liquidity operations when they push the pool imbalance above the threshold.
The draft version of this update is In the stable-surge-disable-unbalanced-liquidity branch.
This is applied in commit 4680d59

Fix Review: the fix includes a partial refactoring of the hook contract and adds checks afterAddLiquidity and afterRemoveLiquidity to block unbalanced liquidity operations if they cause the pool to surge. It prevents the described issue and does not introduce new problems.

L-02 Fees are not included in the threshold or fee calculations		
Severity: **Low**	Impact: **Low**	Likelihood: **likely**
Files: StableSurgeHook.sol	Status: Will not be fixed	

Description: The onComputeDynamicSwapFeePercentage hook checks if a swap imbalances the pool above the threshold and increases the fee percentage when this is the case. A part of this fee can be added to the pool's liquidity and can cause the pool to imbalance more. This creates a circular dependency between the fee and the imbalance. The fee imbalances the pool more, the higher imbalance should cause a higher fee, which would imbalance the pool more, etc.
Additionally the imbalance threshold is calculated on the swap parameters before swap fees are subtracted. This can trigger the hook to charge additional fees, while the imbalance of the pool after the swap would remain under the threshold if fees were taken into account.

Not taking the fees in account for the calculations causes the user to pay slightly more or slightly less fees then what he would pay when fees are included in the calculations.

Recommendations: Consider including the fees in the calculations

Customer's response: Balancer is aware of this effect. Including fees in all calculations would make the calculations significantly more complex. Core intention of the SurgeHook is to charge a higher fee when a swap is pushing the pool's imbalance above a threshold. Having a small error margin for the calculated fees or threshold is acceptable as long as the fees remain above the static swap fee, cannot be bypassed and can not cause the pool to lose.
The team is open to suggestions that can improve the calculations with little extra complexity to the code.

L-03 Incorrect maxSurgeFeePercentage setting can cause underflow

Severity: **Low**	Impact: **Medium**	Likelihood: **low**
Files: StableSurgeHook.sol	Status: Fixed in 767a6a1	

Description: The maxSurgeFeePercentage setting in StableSurgeHook is used to specify the maximum fee percentage that is charged then the pool is completely unbalanced.
The pool swapFeeManager can change the maxSurgeFeePercentage value to any value between 0 and 100% (FixedPoint.ONE).

Setting the maxSurgeFeePercentage smaller than the staticFeePercentage will cause certain swaps to revert due to arithmetic overflow.

When the pool's imbalance is above the threshold, the dynamic swap fee is calculated with

```
Unset
staticFeePercentage +
(surgeFeeData.maxSurgeFeePercentage - staticFeePercentage).mulDown(
       (newTotalImbalance - surgeFeeData.thresholdPercentage).divDown(
       uint256(surgeFeeData.thresholdPercentage).complement()
       )
);
```

Here it can be seen that when surgeFeeData.maxSurgeFeePercentage is smaller than staticFeePercentage, the (surgeFeeData.maxSurgeFeePercentage - staticFeePercentage) part will underflow.

Recommendations: Add checks that maxSurgeFeePercentage is not smaller than staticFeePercentage before doing the calculation and in that case return the staticFeePercentage.

Customer's response: Acknowledged and fixed in [767a6a1]

Fix Review: Applied fix resolves the issue

Informational Severity Issues

I-01. Users can front-run swaps and force surge pricing for other users.

The calculation of imbalance checks for a total shift of imbalance when determining the new fee surge price. Due to the fact that only total imbalance is taken into account, if a user tries to bring the pool back into balance they can be front-run in such a way that the swap would now increase the imbalance.

Exploit Scenario:

1. Bob notices that a pool with two coins A and B is now imbalanced.
2. The pool has 10 of coin A and 1 of coin B,
3. Bob swaps 2 A for 2 B.
4. Eve front runs the call and swaps 8A for 8B changing the pool now to 2A and 9B which is more balanced so Eve only pays static fees.
5. Bob Swap now attempts to push the pool for a greater imbalance and so he pays a huge surge price.

Recommendations: Allow users to specify if they expect the swap to accrue dynamic or static fees and revert if that is not the case

Customer's response: Acknowledged, would not fix. This vector is stopped by properly setting tight slippage bounds.

Fix Review: Not applicable.

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.