Three Sigma

# Code Audit

## Liquity CDP Stablecoin Protocol

# Disclaimer

Code Audit

**Liquity** CDP Stablecoin Protocol

# Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program.

# Table of Contents

Code Audit

**Liquity** CDP Stablecoin Protocol

Table of Contents

# Summary

Code Audit

**Liquity** CDP Stablecoin Protocol

# Summary

Three Sigma was approached by the Liquity protocol to perform a review of the fixes implemented to mitigate a critical vulnerability within Stability Pools, reported by ChainSecurity on February 18th.

A detailed analysis of the Stability Pool vulnerability created by ChainSecurity was provided and further analyzed internally.

The issue stemmed from an overcompensation in rounding error corrections, which allowed attackers to manipulate liquidations and withdraw excess funds. To address this, Liquity removed the flawed error correction logic, improved precision calculations, and ensured rounding always favors the Stability Pool, preventing further exploits.

## Protocol Description

Liquity V2 is a decentralized borrowing protocol that enables users to obtain loans by collateralizing Ether (ETH) and liquid staking tokens (LSTs). A key innovation of V2 is the introduction of user-set interest rates, granting borrowers greater control over their borrowing costs. The protocol also introduces BOLD, a stablecoin designed to provide sustainable, real yield to depositors. BOLD maintains its peg through direct redeemability and benefits from improved peg dynamics.

# Scope

Code Audit

**Liquity** CDP Stablecoin Protocol

# Scope

| Filepath | nSLOC |
|---|---|
| contracts/src/StabilityPool.sol | 302 |
| contracts/src/TroveManager.sol::batchLiquidateTroves() | 42 |
| contracts/src/CollateralRegistry.sol::_updateLastFeeOpTime() | 7 |

## Assumptions

Out of scope contracts are considered secure.

# Project Dashboard

## Code Audit

**Liquity** CDP Stablecoin Protocol

# Project Dashboard

## Application Summary

| | |
|---|---|
| Name | Liquity |
| Commit | f87639d |
| Language | Solidity |
| Platform | Ethereum |

## Engagement Summary

| | |
|---|---|
| Timeline | 27/03/2025 to 05/04/2025 |
| № of Auditors | 2 |
| Review Time | 2 person weeks |

# Analysis

## Code Audit

## **Liquity** CDP Stablecoin Protocol

# Analysis

## Issue Overview

The Stability Pool corrects rounding errors to maintain accuracy, but under certain conditions, this correction overcompensates, breaking solvency. An attacker can exploit this flaw by manipulating liquidations and index scaling, ultimately withdrawing more than they should.

This occurs because the error correction does not always behave as expected, sometimes causing the system to overestimate its own funds. The solvency invariant states that total pool deposits must always cover outstanding obligations. However, when the error correction is added to P during liquidation, it can artificially inflate the new index value (Pn), making the system believe it holds more funds than it actually does.

To prevent this over-crediting, the system subtracts 1 wei from P, which usually neutralizes small errors under normal conditions.

However, when P is scaled down (P < 1e9), the error correction is scaled up by 1e9, while the protection remains at -1 wei, making it ineffective against large over-crediting. As a result, an attacker can exploit rescaling to claim significantly more collateral than they should.

## Fix Implementation Summary

The issue was resolved by refining precision calculations, removing unnecessary functionality, and ensuring the Stability Pool remains operational under various edge cases. Key changes include:

- Removing the error correction logic.
- Increasing P precision from 1e18 to 1e36 while maintaining a 1e9 scale factor, achieving a 3x precision increase.
- Ensuring rounding always favors the Stability Pool.
- Removing epoch functionality.
- Introducing a 1 BOLD minimum requirement, enforced in both the Stability Pool and TroveManager.

## Breakdown of PRs

### PR #4: Fix Stability Pool Insolvency Bug

In PR #4, the error correction logic was entirely removed from the Stability Pool, significantly simplifying compounded deposit, collateral, and yield gains calculations. The precision of the running product P was increased from 1e18 to 1e36, which, in combination with the SCALE_FACTOR of 1e9, led to a 3x precision improvement.

The code is now cleaner and significantly more simplified compared to the original version, which included error correction. The removed error correction, while theoretically beneficial for users, was never impactful in terms of its USD value and was not mandatory in the first place.

### PR #14: Scalability Precision Issue

During PR #4 development, a precision issue was detected in the offset() function when calculating the new P value in situations where P falls below 1e27 and requires scaling. This issue was correctly addressed, ensuring proper precision during rescaling.

### PR #9: 1 BOLD Minimum Stability Pool Balance

This PR introduced a minimum deposit requirement of 1 BOLD in the Stability Pool. This prevents the Stability Pool from being completely depleted and ensures that the running product P does not reach zero, preserving proper reward calculations.

### PR #10: Removed Epoch Functionality

The system was simplified by eliminating epochs. With the introduction of the 1 BOLD minimum Stability Pool balance, the running product P will never reach zero directly. As a result, maintaining the epoch functionality was no longer necessary, and its removal further simplified the Stability Pool contract.

### PR #12: Fix Base Rate Decay Can Be Slowed Down

This fix ensures that the base rate decay mechanism properly accounts for time intervals. Previously, calling the function at 1:59 would result in it being counted as only one minute,

causing 59 seconds of decay to be lost. Now, after an extra second, the second interval will be correctly accounted for.

## Conclusion

The fixes implemented have successfully resolved the vulnerability, simplifying the Stability Pool logic while improving its robustness. The removal of error correction and epoch functionality has streamlined the system, and precision adjustments ensure accurate calculations.

One remaining consideration is a potential UI/UX issue related to dust amounts (less than 1 BOLD) left for users after full liquidation and before any new Stability Pool deposit. While this is by design, it could cause confusion at the frontend level.

**Recommended Fix**

- Modify the UI/UX to handle this case appropriately.
- When calling `withdrawFromSP()` in this specific scenario, users should call it with an amount of 0 to successfully claim their collateral, even if a dust amount remains.

The issue has been fully fixed across all PRs.

A minor UI/UX improvement is suggested to enhance user experience.