code4rena

Solera

Smart Contract Security Assessment

Version 1.0

Audit dates: Dec 16 — Dec 18, 2024

Audited by: defsec
said

Contents

1. Introduction
   - 1.1 About Zenith
   - 1.2 Disclaimer
   - 1.3 Risk Classification
2. Executive Summary
   - 2.1 About Solera
   - 2.2 Scope
   - 2.3 Audit Timeline
   - 2.4 Issues Found
3. Findings Summary
4. Findings
   - 4.1 Medium Risk
   - 4.2 Low Risk
   - 4.3 Informational

1. Introduction

1.1 About Zenith

Zenith is an offering by Code4rena that provides consultative audits from the very best security researchers in the space. We focus on crafting a tailored security team specifically for the needs of your codebase.

Learn more about us at https://code4rena.com/zenith.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2. Executive Summary

2.1 About Solera

Solera is a decentralized lending platform aiming to reshape the DeFi landscape on Plume Network with its innovative, highly efficient lending platform, designed for both crypto assets and tokenized RWAs, paving the way for advanced yield and liquidity solutions.

2.2 Scope

Repository	SoleraMarkets/solera-contracts
Commit Hash	02f8b39190e1dbbe4acd0b08a686642a0e28ce78

2.3 Audit Timeline

DATE	EVENT
Dec 16, 2024	Audit start
Dec 18, 2024	Audit end
Dec 30, 2024	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	1
Low Risk	1
Informational	4
Total Issues	6

3. Findings Summary

ID	DESCRIPTION	STATUS
M-1	Lack of initial deposit in the new market listing script	Resolved
L-1	Implementation of consistent methodology for setting supply and borrow caps	Acknowledged

I-1	Upgrade privileged roles to multi-sig after the deployment	Acknowledged
I-2	Incomplete market listings and oracle updates	Acknowledged
I-3	Implement zero approval step for USDT in reward configuration	Resolved
I-4	`ConfigureReward` repeatedly uses hardcoded addresses	Resolved

4. Findings

4.1 Medium Risk

A total of 1 medium risk findings were identified.

[M-1] Lack of initial deposit in the new market listing script

Severity: Medium	Status: Resolved

Context:

- SolV3Listing.sol

Description: The new market listing script does not include or bundle a small initial deposit. This could make the new market prone to the known empty market exploit.

Recommendation: On the mainnet, consider always bundling the listing with a small initial deposit to avoid the empty market issue. For reference, this is the Aave's deployment transaction : here.

Solera: Fixed with the following commit @3bf133902bf12..

Zenith: Verified.

4.2 Low Risk

A total of 1 low risk findings were identified.

[L-1] Implementation of consistent methodology for setting supply and borrow caps

Severity: Low	Status: Acknowledged

Context:

- SolV3Listing.sol#L63

Description:

In the market parameters, supply and borrow caps are critical risk management parameters that help maintain the protocol's stability and security. These caps are designed to limit the amount of an asset that can be supplied or borrowed, thereby mitigating risks such as price exploits and protocol insolvency.

However, many assets currently have their supply and borrow caps set to "0", indicating that these parameters have not been fully implemented or optimized. This situation necessitates the development of a transparent and consistent methodology for setting these caps, especially in light of recent market volatility and changes in sentiment.

The current state of supply and borrow caps in the Solera is inconsistent, with many assets lacking defined caps. The absence of a standardized methodology for setting these caps further complicates the issue, as it leaves the protocol exposed to risks associated with market fluctuations.

Recommendation: Adopt the outlined methodology for setting supply and borrow caps across all assets in the deployment.

Solera: Acknowledged - All markets will be initialized with supply and borrow caps, the sheet will be updated before deployment and we will review the initialization txns together here before they go out

4.3 Informational

A total of 4 informational findings were identified.

[I-1] Upgrade privileged roles to multi-sig after the deployment

Severity: Informational	Status: Acknowledged

Context:

- DefaultMarketInput.sol

Description: In the `DefaultMarketInput` contract, the market owner, emergency admin, and pool admin roles are currently assigned to a single deployer address.

This configuration centralizes control, increasing the risk of unauthorized access or misuse. By transitioning these roles to a multi-sig wallet, the protocol can distribute control among multiple parties, reducing the risk of single-point failures.

Recommendation: Transition the market owner, emergency admin, and pool admin roles to multi-sig wallets. This will require multiple approvals for critical actions.

Solera: Acknowledged - We will deploy with a multisig address during deployment.

[I-2] Incomplete market listings and oracle updates

Severity: Informational	Status: Acknowledged

Context:

- 1_UpdateOracles.sol#L45

Description:

Various markets are defined with specific parameters, including PLUME, sPLUME, ETH, plumeETH, rswETH, STONE, nRWA, nTBILL, nYIELD, USDT, USDC, and pUSD. These markets are crucial for the protocol's functionality, providing users with options for lending, borrowing, and trading. However, the current implementation of the `UpdateOracles` contract only includes a few of these markets, specifically WETH, USDT, and USDC.e, for oracle updates.

Recommendation:

Modify the `UpdateOracles` contract to include all defined markets in the price feed updates.

Solera: Acknowledged - Will add missing token prices oracles during mainnet deployment + use real Stork Oracles when price feeds are deployed by Stork on Mainnet

[I-3] Implement zero approval step for USDT in reward configuration

Severity: Informational	Status: Resolved

Context:

- 2_ConfigureReward.sol#L31

Description: The `ConfigureReward` contract script is responsible for setting up reward configurations, including the approval of USDT tokens for transfer strategies.

The current implementation directly approves a specified amount of USDT without first setting the allowance to zero. This line directly sets the allowance for the transferStrategy contract. However, some ERC20 tokens, including USDT, may require the allowance to be set to zero before it can be updated to a new value.

Recommendation: Modify the script to first set the USDT allowance to zero before setting it to the desired amount.

Solera: Fixed with the following commit @7f624b16db6a5..

Zenith: Verified

[I-4] `ConfigureReward` repeatedly uses hardcoded addresses

Severity: Informational	Status: Resolved

Context:

- 2_ConfigureReward.sol#L26-L27
- 2_ConfigureReward.sol#L30
- 2_ConfigureReward.sol#L36-L37
- 2_ConfigureReward.sol#L46-L47

Description: To improve script readability and minimize errors, the referenced line can avoid using repeated hardcoded addresses.

Recommendation: Use the addresses already defined in `AaveV3PlumeAssets` for `ConfigureReward` asset addresses. Additionally, use the predefined `ACL_ADMIN` as the emission admin for `PullRewardsTransferStrategy` instead of hardcoding the address.

Solera: Fixed with the following commit @caa32425965576..

Zenith: Verified