PUBLIC

Code Assessment

of the Sulu Extensions Smart Contracts

March 07, 2024

Produced for

by

Contents

1 Executive Summary

Dear Mona and Sean,

Thank you for trusting us to help Avantgarde Finance with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Sulu Extensions according to Scope to support you in forming an opinion on their security risks.

Avantgarde Finance implements extensions for Sulu. These extensions concern the implementation of a universal Curve adapter, the redemption of deprecated synthetic tokens for sUSD, improvements that allow users to repay their full Compound debt positions, the configuration of the name and the symbol of the shares of the funds, and the interaction with Olympus protocol.

During the review, no important issues were uncovered. All the minor issues have been fixed. The most critical subjects covered in our audit are functional correctness, access control and precision of arithmetic operations. Security regarding all the aforementioned subjects is high. General subjects covered were code complexity, gas efficiency, documentation and specification. All the aforementioned subjects were of high quality.

In summary, we find that the codebase provides a high level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	2
• Code Corrected	2

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

The assessment was performed on the source code files inside the Sulu Extensions repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	20 December 2021	e6e1ede6acc7a9259afbb0efbaf0e7ecfef1c250	Initial Version of Extensions
2	06 January 2022	296c1e41c7dd59ae9fbaa21b4d4e68c6234e6217	Second Version of Extensions

For the solidity smart contracts, the compiler version `0.6.12` was chosen.

The following files were considered in scope:

- release/extensions/integration-manager/integrations/adapters/CurveLiquidityAdapter.sol
- release/extensions/integration-manager/integrations/utils/actions/CurveLiquidityActionsMixin.sol
- release/extensions/integration-manager/integrations/adapters/SynthetixAdapter.sol
- release/extensions/integration-manager/integrations/utils/actions/SynthetixActionsMixin.sol
- persistent/vault/VaultLibBase2.sol
- release/core/fund-deployer/FundDeployer.sol
- release/core/fund/vault/IVault.sol
- release/core/fund/vault/VaultLib.sol
- release/extensions/integration-manager/integrations/adapters/OlympusV2Adapter.sol
- release/extensions/integration-manager/integrations/utils/actions/OlympusV2ActionsMixin.sol
- r/e/external-position-manager/external-positions/compound-debt/CompoundDebtPositionLib.sol
- r/e/external-position-manager/external-positions/compound-debt/CompoundDebtPositionParser.sol

2.1.1 Excluded from scope

The rest of the contracts that are part of Sulu.

2.2 System Overview

This system overview describes the initially received version (Version 1) of the contracts as defined in the Assessment Overview.

Furthermore, in the findings section, we have added a version icon to each of the findings to increase the readability of the report.

Avantgarde Finance implements the following extensions/updates to Sulu:

- allow the SynthetixAdapter to redeem the deprecated synths to sUSD
- make shares token name and symbol freely settable
- a universal liquidity adapter for Curve
- a staked OHM adapter
- allow to repay the CompoundDebtPosition in full

2.3 SynthetixAdapter

The synthetix adapter is extended so it allows for the redemption of deprecated synths for sUSD. Fund managers can call `SynthetixAdapter.redeem` which calls `redeemAll` of the `SynthRedeemer`.

2.4 Setting Token Name and Symbol

Fund managers are now allowed to set and update the token symbol for their funds. Both name and symbol can be defined during the creation of the fund and changed via `VaultLib.setName/setSymbol` by the owner of the fund.

2.5 Universal Curve Liquidity Adapter

This adapter takes advantage of the common interface among Curve pools. Fund managers are allowed to:

- add liquidity to a pool,
- stake the LP tokens and receive Gauge tokens,
- add liquidity and directly stake,
- unstake their LP tokens,
- withdraw the liquidity either in one or all coins of the pool by giving back LP tokens,
- unstake and withdraw,

The adapter constructs the call to the pool appropriately so that managers can also deposit the underlyings of tokens of the pools. For example, a manager can deposit the underlying token in a pool of Aave tokens.

2.6 Staked OHM Adapter

The adapter allows managers to stake or unstake `OHM` or `sOHM` tokens respectively.

2.7 Repaying full Compound Debt Position

Funds are now allowed to pass max uint256 to repay the entire debt amount instead of overestimating it.

2.7.1 Trust Model

The admins of Curve registry are assumed to not act maliciously and update the registry with the correct address for the LP tokens of the pools. Moreover, the synthetix redeemer is assumed to work correctly and, therefore, there is no need for Sulu to use slippage protection by setting the minimum expected value of sUSD to be received upon redemption.

3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

4 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- *Likelihood* represents the likelihood of a finding to be triggered or exploited in practice
- *Impact* specifies the technical and business-related consequences of a finding
- *Severity* is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

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

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

5 Findings

In this section, we describe any open findings. Findings that have been resolved have been moved to the Resolved Findings section. The findings are split into these different categories:

- Design: Architectural shortcomings and design inefficiencies

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

6 Resolved Findings

Here, we list findings that have been resolved during the course of the engagement. Their categories are explained in the Findings section.

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	2

- Redundant While-loop Code Corrected
- Unstaking sOHM Leaves Dust Code Corrected

6.1 Redundant While-loop

Design Low Version 1 Code Corrected

`curveLiquidityAdapter.__parseSpendAssetsForLendingCalls` aims to detect the spent assets and the amounts of them so that they can be post-processed by the Integration Manager. It includes the following while-loop which should always terminate after one iteration.

```
while (spendAssetsIndex < spendAssetsCount) {
    for (uint256 i; i < _orderedOutgoingAssetAmounts.length; i++) {
        if (_orderedOutgoingAssetAmounts[i] > 0) {
            spendAssets_[spendAssetsIndex] = __castWrappedIfNativeAsset(
                canonicalPoolAssets[i]
            );
            spendAssetAmounts_[spendAssetsIndex] = _orderedOutgoingAssetAmounts[i];
            spendAssetsIndex++;
        }
    }
}
```

Notice that `spendAssetsIndex` increases to the maximum value of `spendAssetCount` inside the for-loop.

Code corrected:

The while-loop has been removed while an early exit of the for-loop when `spendAssetsIndex == spendAssetsCount` was added.

6.2 Unstaking sOHM Leaves Dust

Design Low Version 1 Code Corrected

`sOHM` is a rebasing token, meaning that the number of tokens a vault has increases after each epoch. Sulu currently allows managers to define the number of `sOHM` tokens they want to unstake. Consider the following case:

- A fund manager `M` wants to unstake all the `sOHM` the fund holds
- The fund manager submits a transaction where they unstake the total amount of the tokens
- A rebase happens and the number of `sOHM` increases
- The transaction is mined.

This will lead to the fund holding a dusty amount of `sOHM` together with the unstaked `OHM` tokens.

Code corrected:

The code has been adapted to support unstaking the maximum amount when `uint.max` is specified as the unstake amount.

7 Notes

We leverage this section to highlight further findings that are not necessarily issues. The mentioned topics serve to clarify or support the report, but do not require an immediate modification inside the project. Instead, they should raise awareness in order to improve the overall understanding.

7.1 Rewards in Different Gauge Version

Note Version 1

`CurveLiquidityAdapter` is implemented in such a way so that it is compatible with version 2, 3 and 4 of the gauge tokens. When `claim_rewards` for the gauge tokens v2 and v3, rewards are accrued. However, this is not true for v4 where users should pass a specific argument for the rewards to be accrued.