PUBLIC

Code Assessment

of the Sulu Extensions XIII Smart Contracts

September 25, 2023

Produced for

by

Contents

1 Executive Summary

Dear Enzyme team,

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 XIII according to Scope to support you in forming an opinion on their security risks.

Avantgarde Finance implements changes and extensions on the Sulu system. In particular, the changes consist of refactoring and bug fixes of the Aura/Convex staking adapter, a new policy that prevents redemption in specific assets when some assets are depegged, the introduction of the ArrakisV2 adapter, the extension of the deposit wrapper so that users can exchange an arbitrary asset to the denomination asset of the fund of which they want to buy shares, the AaveV3 CDP external position and the Lido stETH withdrawal.

The most critical subjects covered in our audit are asset solvency, functional correctness, front-running, and accurate fund valuation. No major issues were uncovered.

The general subjects covered are code complexity, upgradeability, unit testing, and documentation. The security of all aforementioned subjects is high.

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	1
• Risk Accepted	1

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 XIII 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	11 Sep 2023	bde5f7e5bd30caad1965599f5151daf31bd138c9	Initial Version
2	25 Sep 2023	ac98ee0ef85701b15b127b52cf5e785fd0851ab4	Fixes

For the solidity smart contracts, the compiler version `0.8.19` was chosen.

The smart contracts in scope are:

2.1.1 Convex/Aura staking wrapper changes

`extensions/integration-manager/adapters/`

- `AuraBalancerV2LpStakingAdapter.sol`
- `ConvexCurveLpStakingAdapter.sol`
- `utils/0.6.12/actions/StakingWrapperActionsMixin.sol`

`infrastructure/price-feeds/derivatives/feeds`

- `ConvexCurveLpStakingWrapperPriceFeed.sol`
- `staking-wrappers/`:
  - `aura-balancer-v2-lp`:
    - `AuraBalancerV2LpStakingWrapperFactory.sol`
    - `AuraBalancerV2LpStakingWrapperLib.sol`
  - `convex-curve-lp/`:
    - `ConvexCurveLpStakingWrapperFactory.sol`
    - `ConvexCurveLpStakingWrapperLib.sol`
  - `StakingWrapperBase.sol`
  - `StakingWrapperLibBase.sol`

`utils/0.8.19/`:

- `dispatcher-owned-beacon/DispatcherOwnedBeacon.sol`

It is important to note that the changes here were part of a refactoring. During the review it was assumed that the previously implemented logic as far as the interaction with the external systems is concerned was correct.

2.1.2 ArrakisV2 Adapter

`extensions/integration-manager/adapters/ArrakisV2Adapter.sol`

`infrastructure/price-feeds/derivatives/feeds/ArrakisV2PriceFeed.sol`

`utils/0.8.19/`:

- `adapted-libs/`:
  - `CorePositionValue.sol`
  - `LiquidityAmounts.sol`
- `UniswapV3PositionHelper.sol`

The logic implemented by `CorePositionValue`, `LiquidityAmounts`, `UniswapV3PositionHelper` was only compared to the respective UniswapV3's implementation. Moreover, the valuation of Arrakis vault shares was only compared with the logic implemented by ArrakisV2 and the respective UniswapV3 implementation.

2.1.3 No Active Depeg during Redemption Policy

`extensions/policy-manager/asset-managers/`:

- `current-shareholders/NoDepegOnRedeemSharesForSpecificAssetsPolicy.sol`
- `0.8.19/`:
  - `NoDepegPolicyBase.sol`
  - `PolicyBase.sol`

2.1.4 ERC20 Exchange and Buy

`core/fund/comptroller/peripheral/DepositWrapper.sol`

2.1.5 AaveV3 CDP External Position

`extensions/external-position-manager/external-positions/aave-v3-debt`:

- `AaveV3DebtPositionDataDecoder.sol`
- `AaveV3DebtPositionLib.sol`
- `AaveV3DebtPositionParser.sol`

`persistent/external-positions/aave-v3-debt/AaveV3DebtPositionLibBase1.sol`

2.1.6 LidoWithdrawals External Position

`persistent/external-positions/lido-withdrawals/LidoWithdrawalsPositionLibBase1.sol release/extensions/external-position-manager/external-positions/lido-withdrawals`:

- `LidoWithdrawalsPositionDataDecoder.sol`
- `LidoWithdrawalsPositionLib.sol`
- `LidoWithdrawalsPositionParser.sol`

2.1.7 Excluded from Scope

All the contracts that are not explicitly mentioned in the Scope section are excluded from scope. The external systems, with which the Sulu interacts, are assumed to work correctly. Moreover, all the libraries used such as the OpenZeppelin library are assumed to work correctly and not in scope of this review. Finally, attack vectors, such as unfavourable for the fund trades, employed by the managers of the funds have not been considered as the managers are considered trusted by the system.

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.

This assessment is only concerned with specific code parts of the system. Here, we only provide a brief overview of the newly introduced modules of the system. Please refer to Enzyme's documentation or our previous reports for a more detailed overview of the system.

Enzyme is an investment management system that allows one to buy shares of certain vaults. Vaults act like funds and each vault has a vault manager, who is in charge of investing the funds of the vault into other DeFi protocols. The manager can set certain policies that limit the actions of the users including the manager's. Furthermore, the manager can only invest in third-party protocols for which an adapter exists. Avantgarde Finance expands the system by adding new policies, adapters, and external positions to expand and improve the investment opportunities for the vault managers.

The assessment was performed on the following new, refactored, and patched functionalities which integrate into Enzyme Finance's overall ecosystem:

- A fix in the `StakingWrapperBase` contract's reward calculation logic.
- A refactoring to improve the Convex/Aura staking wrapper contracts.
- A new policy that monitors for a depeg of any stable asset that the fund specifies in the policy.
- An Arrakis v2 LP integration that is primarily for the benefit of allowing vaults to hold Uni v3 positions as ERC20 tokens.
- A wrapper that adds an exchange-and-buy-shares option for ERC20 tokens for users.
- A new integration for Aave v3 CDPs.
- A new integration for Lido sETH withdrawals.

The system overview is divided into multiple sections that will provide a brief description of each of the code components above.

2.2.1 Fix: StakingWrapperBase reward calculations

`StakingWrapperBase.updateHarvestAndClaim()` should always write `lastCheckpointBalance`, but only do so if the balance after payout to the user is less than the previously-checkpointed balance. This behavior was changed to always write the `lastCheckpointBalance` by removing the condition `finalBal < totalHarvestData.lastCheckpointBalance` before writing `totalHarvestData.lastCheckpointBalance = uint128(finalBal);`.

2.2.2 Refactor: Convex/Aura staking wrapper improvements

Convex and Aura are two integrations sharing the same staking wrapper code. Besides some minor changes and the compiler update, the following changes were made:

- Factory and proxies now use OpenZeppelin's beacon proxies.
- Decoupling of withdrawing and claiming rewards including a new checkpoint. To prevent withdrawals from getting struck.
- The change before makes the removal of bad reward tokens unnecessary. Hence, the logic to remove bad tokens and catch errors while harvesting was removed.
- Pausing the `__harvestRewardsLogic` logic was removed from `__checkpointAndClaim` and `__checkpoint`. Both will always trigger the harvest logic now. Pausing only applies to deposits.
- Redundant functionality was removed from withdraw and deposit logic.

2.2.3 New Policy: No active depeg during redemption

This policy monitors for a depeg of any stable asset that the fund specifies in the policy (it doesn't crawl the fund's holdings). It applies to any asset with a reference invariant like USDC/USDT/DAI (USD) but also stETH (ETH).

2.2.4 New integration: Arrakis v2 LP

The new Arrakis v2 LP integration allows vaults to hold Uni v3 positions as ERC20 tokens (i.e., not as an NFT in an external position). All Arrakis vaults will be carefully added to the asset universe by the Enzyme Council, only if determined that the manager role for that vault is trustless (or the manager is suitable to be trusted by all funds).

The Arrakis integration primarily consists of the adapter (`ArrakisV2Adapter`) and the price feed (`ArrakisV2PriceFeed`). The adapter contract allows minting shares in an Arrakis vault by calling `lend` and burning Arrakis vault shares by calling `redeem`. Otherwise, the adapter parses and processes the call arguments to handle the calls similar to existing adapters. The price feed contract evaluates the value of the Arrakis vault shares by replicating Arrakis's price evaluation. The evaluation checks the price of the underlying assets in the corresponding Uniswap pool, adds the earned fees, and the value of the assets stored in the Arrakis vault and deducts the Arrakis fees and the manager's balance in the Arrakis vault.

2.2.5 DepositWrapper: Add erc20 exchange-and-buyShares option

The `DepositWrapper` contract allows swapping EC20 tokens on an arbitrary exchange and buying vault shares for the swapped tokens.

2.2.6 New integration: Aave v3 CDPs

The new integration uses Enzyme's external integration logic. It allows fund managers to use Aave v3 CDPs. The fund manager can use the integration for six different actions:

- Add collateral assets: based on the arguments, this action either wraps some underlying assets into aTokens by calling `supply()` on the Aave pool or transfers some a tokens from the vault to the external position.
- Remove collateral assets: based on the arguments, this action either unwraps some aTokens by calling `withdraw()` on the Aave pool or simply transfers aTokens to the vault.
- Borrow assets: Borrow assets from Aave (call `borrow()` on the Aave pool) and transfer those assets to the vault.
- Repay assets: Withdraw some borrowed underlyings from the vault to repay a loan on Aave by calling `repay()` on the pool.
- Set EMode: sets the EMode for the external position

- Set an asset as collateral: allows the user to set or unset an asset as collateral.

2.2.7 New integration: Lido stETH withdrawals

External position for requesting and claiming Lido stETH exits. The external position needs to manage the state of the withdrawal requests as there's a cool-down period between requesting and actually claiming. The position implements two actions:

- Make a new withdrawal request. This creates a new request which is stored in the external position.
- Claim a withdrawal using a request. The claimed ETH is sent to the vault and the request is deleted.

2.2.8 Roles and Trust Model

Please refer to the main audit report and the extension audit reports for a general trust model of Sulu.

No new roles are introduced by the components under review.

The actions of the `owner` of the adapters are slightly modified. More specifically, the `owner` can now toggle pausing in the `StakingWrapperBase`. If pausing is enabled, no depositing is possible anymore. The `ConvexCurveLPStaking` also allows the owner to call `pauseWrappers` and `unpauseWrappers` which then can batch call `togglePause` on the wrappers.

Arrakis vaults are fully trusted and assumed to work as intended and they should not be able to harm any vaults of the Sulu system. The same applies to Lido's withdrawal queue and AaveV3 pool.

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	1

- Missing Sanity Check Risk Accepted

5.1 Missing Sanity Check

Design Low Version 1 Risk Accepted

CS-SUL13-001

When the NoDepegePolicy is set, no sanity checks are performed with regard to the asset pairs. This means that a fund owner could wrongfully set assets that are not even part of the supported assets. This will result, in the `redeemSharesForSpecificAssets()` failing as the ValueInterpreter will not be able to process the assets.

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	0

6.1 Redundant Event Defintion

Informational Version 1 Code Corrected

CS-SUL13-003

`StakingWrapperBase` defines the `RewardsClaimed` event which is never emitted.

Code corrected:

The event has been removed.

7 Informational

We utilize this section to point out informational findings that are less severe than issues. These informational issues allow us to point out more theoretical findings. Their explanation hopefully improves the overall understanding of the project's security. Furthermore, we point out findings which are unrelated to security.

7.1 Deleting From Lists

Informational **Version 1**

CS-SUL13-002

Lido stores the pending requests in storage. When a request is claimed, the claim request is deleted from storage. This happens in the following steps. `__claimWithdrawals()` iterates over all the pending requests until it finds the claimed one. It then replaces it with the last pending request in the `requests` list which is then popped by the end of the list. This procedure is gas-inefficient. Consider the case where the stored requests are iterated in reverse order and the `requestIds` passed as arguments are also in reverse order. Then it's more likely that the claimed withdrawal will be the last one in the stored list meaning that no iteration steps through the list are needed. Moreover, only one pop operation is needed to delete the request without any replacement.

8 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.

8.1 Adding Collateral of Same Type

Note Version 1

When using the AaveV3 external position, a manager can only supply either underlying tokens or aTokens. For example, if the manager wants to supply aUSDC and DAI, they have to make two separate calls to the external position.