PRIVATE

Code Assessment

of the Sulu Extensions Smart Contracts

December 7, 2021

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.

The present report covers the implementation of the extension for UniswapV3-LP, the adapters for PoolTogetherV4 and ParaswapV5, the list attestation for the Address List Registry as well as the Fund Value Calculator and its wrappers. All in all, only minor issues were uncovered which were addressed.

The most critical subjects covered in our audit are functional correctness and access control. Security regarding all the aforementioned subjects is high.

The general subjects covered are upgradability, unit testing and gas efficiency. Security regarding all the aforementioned subjects is high. The specification provided was comprehensive.

In summary, we find that the aforementioned modules to be added to the system of 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	3
• Code Corrected	3

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	15 November 2021	b413077c042511f37f8c025603dc8a806a8b7031	Initial Version for v4
2	15 November 2021	828c32e33a84604001ccb7759bfb0afc4f3ce6a3	Initial Version for v3
3	06 December 2021	fec37a7e726f41961c8b769c5796afaa84a53fc0	Second Version for v4

For the solidity smart contracts, the compiler version `0.6.12` was chosen, for the smart contracts related to UniswapV3 LP the compiler version `0.7.6` was chosen. More specifically, the files in scope are the following:

- persistent/off-chain/fund-value-calculator-usd-wrapper/FundValueCalculatorUsdWrapper.sol
- persistent/off-chain/fund-value-calculator/FundValueCalculatorRouter.sol
- release/off-chain/FundValueCalculator.sol
- [v3] release/off-chain/FundValueCalculator.sol
- release/extensions/integration-manager/integrations/adapters/ParaSwapV5Adapter.sol
- release/extensions/integration-manager/integrations/utils/actions/ParaSwapV5Actions.sol
- release/infrastructure/price-fees/derivatives/feeds/PoolTogetherV4PriceFeed.sol
- release/extensions/integration-manager/integrations/adapters/PoolTogetherV4Adapter.sol
- release/extensions/integration-manager/integrations/utils/actions/PoolTogetherV4Actions.sol
- persistent/external-positions/uniswap-v3-liquidity/UniswapV3LiquidityPositionLibBase1.sol
- release/extensions/external-position-manager/external-positions/uniswap-v3-liquidity/:
    - UniswapV3LiquidityPositionDataDecoder.sol
    - UniswapV3LiquidityPositionLib.sol
    - UniswapV3LiquidityPositionParser.sol
- persistent/address-list-registry/AddressListRegistry.sol

The acknowledged considerations raised in the Sulu report which can be related to the extensions under scope are not repeated. Readers can refer to the Sulu audit report.

2.1.1 Excluded from scope

Any contract not explicitly listed above. Moreover, we consider out-of-scope the implementation of `UniswapV3LiquidityPositionLib.__uniswapSqrt` which is copied from Uniswap.

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 to Sulu:

- dedicated contracts to retrieve metrics for a fund
- an adapter to ParaSwapV5 platform
- an adapter to PoolTogetherV4
- UniswapV3 LP external position
- list attestation for `AddressListRegistry`

2.3 Fund Value Calculator

The `FundValueCalculatorRouter` allows anyone to get metrics of any `VaultProxy`. The module allows for the calculation of the gross asset value (GAV, GAV per share), and the net asset value (NAV, NAV per share, NAV of a given shareholder) i.e., where fees are already deducted from the calculation. Those metrics are available in multiple quote assets: user-specified quote asset, denomination asset of the target `VaultProxy`, or USD.

The user can either call the `FundValueCalculatorRouter` to have access to the first two quote assets options, and `FundValueCalculatorUsdWrapper` to get the values in USD.

Each fund deployer is bound to a `FundValueCalculator` contract.

All the calculations related to the value of the fund do not require the synthetix assets to be settled.

2.4 ParaSwapV5

Authorized fund managers can swap tokens held by the `VaultProxy` with the help of `ParaSwapV5`. The managers will target `ParaSwapV5Adapter.takeOrder`.

The `VaultProxy` will send the outgoing assets to the adapter, which will then call `IParaSwapV5AugustusSwapper.protectedMultiSwap` that will proceed with the swap and send the incoming tokens to the `VaultProxy`.

2.5 PoolTogetherV4

PoolTogetherV4 allows the authorized fund managers to deposit assets in a `PoolTogether`'s pool and potentially win a reward from this pool. The fund managers can choose between 3 actions:

- `PoolTogetherV4Adapter.lend`: make a deposit to a pool and receive the corresponding amount of `Ticket` token (1:1). The adapter will then call `PrizePool.depositToAndDelegate` on PoolTogether's contract, which will mint `Ticket` to the `VaultProxy`. The delegation is important so that the fund is eligible for rewards.

- `PoolTogetherV4Adapter.claimRewards`: claim the reward on a pool, anybody can claim rewards on behalf of other users. This will transfer the reward amount of `Ticket` to the `VaultProxy`.
- `PoolTogetherV4Adapter.redeem`: withdraw an amount of deposited funds or rewards. The `VaultProxy` will approve the amount to the adapter, which will call `PrizePool.withdrawFrom`. This will burn the `Ticket` held by the vault and then the corresponding amount of the underlying asset is sent back to the vault.

The fund managers must check and claim for rewards themselves. The rewards is diluted after 60 days if not claimed.

2.6 Uniswap V3 LP external position

Authorized fund managers can open, close and collect fees from LP positions on Uniswap V3.

- Firstly, the external position must be created. It is made of the two tokens (`token0` and `token1`) the managers want to provide to a pool and will hold the NFTs representing a position on a pool with this token pair. No check whether the pool exists is done yet.
- Once the external position is created, managers can mint a new position on Uniswap V3 by executing `UniswapV3LiquidityPositionLib.__mint`. The vault will send the desired assets to the external position proxy, which will approve the `NonfungiblePositionManager` for the desired amounts and call `NonfungiblePositionManager.mint`. The external position proxy will then receive the NFT that represents the position on the Uniswap's pool.
- Authorized fund managers can also add more liquidity to one of their position by executing `UniswapV3LiquidityPositionLib.__addLiquidity`. The vault will send the desired assets to the external position proxy, which will approve the `NonfungiblePositionManager` for the desired amounts and call `NonfungiblePositionManager.increaseLiquidity`.
- Managers can remove liquidity from a position by executing `UniswapV3LiquidityPositionLib.__removeLiquidity`. The external position will call `NonfungiblePositionManager.decreaseLiquidity` and collect the amount of (`token0`, `token1`), transferring them back to the `VaultProxy`.
- Managers can collect the fee earned by the external position by executing `UniswapV3LiquidityPositionLib.__collect`. The external position will call `NonfungiblePositionManager.collect`, which will send the earned fees to the `VaultProxy`.
- Finally, managers can purge the position by executing `UniswapV3LiquidityPositionLib.__purge`. The `_liquidity` argument can have multiple meanings:
    - `_liquidity == 0`: no liquidity left, only collect from position and burn NFT
    - `0 < _liquidity < type(uint128).max`: amount of liquidity is known, decrease liquidity, collect and burn if position is empty
    - `_liquidity == type(uint128).max`: amount of liquidity is unknown, compute remaining liquidity (gas expensive), decrease liquidity, collect and burn

Any unspent asset will be sent back to the `VaultProxy`.

An external position can hold NFTs from different pools, as long as the tokens' pair is the same.

2.7 List Attestation

The interface of the `AddressListRegistry` is expanded so that it allows the owner of the lists to emit an event on the blockchain which describes the purpose of the list. |

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 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 associated with the items defined in the engagement letter on whether it is used in accordance with its specifications by the user meeting the criteria predefined in the business specification. 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 the facts that over the life cycle of any software product changes to the product itself or to its environment, in which it is operated, can have an impact leading to operational behaviours other than 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 severities. 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	3

- Code With No Effect Code Corrected
- Missing Sanity Check Code Corrected
- Redundant Deadline Code Corrected

6.1 Code With No Effect

Design Low Version 1 Code Corrected

In `UniswapV3LiquidityPositionLib.__mint`, `token0` and `token1` are overwritten by their own value, hence this code has no effect.

Code corrected: The related assignments have been removed.

6.2 Missing Sanity Check

Design Low Version 1 Code Corrected

Upon UniswapV3 LP external position creation, sorting of `token0` and `token1` is not enforced. UniswapV3 NonfungiblePositionManager needs the tokens to be sorted (`address(token0) < address(token1)`), otherwise the transaction will revert. This means that a non-functional external position can be instantiated. For example, passing non-sorted tokens to NonfungiblePositionManager.mint will revert on PoolAddress.computeAddress which requires the tokens to be sorted.

Code corrected:

Ordering of the tokens is now enforced on the external position initialization.

6.3 Redundant Deadline

Design Low Version 1 Code Corrected

When `UniswapV3LiquidityPositionLib._removeLiquidity` is called, the deadline is set to `block.timestamp + 1`. This is not needed since the call to the `NonfungiblePositionManager` is part of an already executing transaction.

Code corrected:

+1 has been removed.

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 PoolTogether V4 Early Exit Fee

Note Version 1

Upon redeem, PoolTogetherV4 is assumed to have no penalty fee for early withdraws as described in https://docs.pooltogether.com/faq/v3-to-v4-differences. An exit fee like in V3 could prevent the fund manager to withdraw from PoolTogetherV4. More specifically, pools that make use of `PrizePool` are assumed get the full amount requested on withdrawal. When the `PrizePool.withdrawFrom` is called, the amount to be redeemed is calculated using `_redeem` internal function. In the case of `PrizePool._redeem`, this function calls one of the yield sources implementations which determines the actual amount to be redeemed. Should such a yield source return a redeemed amount that is less than the amount initially requested, the call on integration will fail.

7.2 Price Oracle Discrepancies

Note Version 1

In order to calculate the value of a position the actual price between the two tokens is required. For this, the default oracles for the two tokens are used. We assume that there are no big discrepancies between the actual price of the Uniswap pool for the specific pair and the price calculated by the system.