PUBLIC

Code Assessment

of the Sulu Extensions X Smart Contracts

May 15, 2023

Produced for

by

Contents

1 Executive Summary

Dear Avantgarde Finance 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 X according to Scope to support you in forming an opinion on their security risks.

Avantgarde Finance extended the functionality of the `ParaSwapV5` adapter to add support for `MegaSwap` and `SimpleSwap` and changed how the errors are caught. Moreover, the functionality of the `GatedRedemptionQueueSharesWrapper` was extended to allow vault owners to force the transfer of shares from one account to another and to enable a "Request" `DepositMode`. Finally, two new adapters were introduced for `ZeroExV4` and `1inch` swaps.

The most critical subjects covered in our audit are functional correctness, access control, and integration with external protocols. One high-severity issue was found in `GatedRedemptionQueueSharesWrapper`, where a user can purposefully front-run a `depositFromQueue` call and make another user who made a deposit request lose their funds. All the issues have been addressed. The general subjects covered are code complexity, upgradeability, and documentation. Security regarding all the 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	1
• Code Corrected	1
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 X 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	2 May 2023	1dc4357f1db54a565e8f93416511123d1101dac8	Initial Version
2	9 May 2023	da8ed1e235f986b23d909cb717d0cc290e346fc6	Final Version

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

The following files are included in the scope:

`persistent/shares-wrappers/gated-redemption-queue/`:

- `GatedRedemptionQueueSharesWrapperLib.sol`
- `GatedRedemptionQueueSharesWrapperFactory.sol`
- `base/GatedRedemptionQueueSharesWrapperLibBase1.sol`

Note that the above contracts have been in scope of the previous review Sulu extensions IX. The contracts have been extended since then. The review was concerned only with the additional functionality introduced.

`release/extensions/integration-manager/integrations/adapters/`:

- `ZeroExV4Adapter.sol`
- `OneInchV5Adapter.sol`
- `ParaSwapV5Adapter.sol`

`release/extensions/integration-manager/integrations/utils/actions/`

- `ZeroExV4ActionsMixin.sol`
- `OneInchV5ActionsMixin.sol`
- `ParaSwapV5ActionsMixin.sol`

Note that `ParaSwapV5Adapter.sol` and `ParaswapV5ActionMixin.sol` were initially reviewed in Sulu extensions I. These contracts were audited only with regards to the additional functionality introduced.

2.1.1 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.

Avantgarde Finance implements integrations with various external systems. Some integrations were newly added, others were updated. Additionally, the GatedSharesWrapper was updated.

2.2.1 GatedSharesWrapper: Add forced transfers and Request DepositMode

For a full description of the shares wrapper, consult the Sulu Extensions IX audit report.

The GatedSharesWrapper has been updated to include three new features:

1. `forceTransfer()` allows the vault owner to move wrapped shares from any address to any other address without their approval. Unprocessed deposit requests are unaffected. This is intended to be used in case of loss of private keys. Note that this means the owner is fully trusted and can take control of any user's wrapped shares at any time.
2. The wrapper can use the native asset as a redemption asset. In this case, `kick` sends a wrapped version of the asset to its recipient, and `redeemFromQueue` unwraps the asset before it sends it to the recipient.
3. A new DepositMode has been introduced, which can be enabled to give the wrapper managers control over when to process deposits instead of allowing them at any time.

The following functions are used with the new Request DepositMode:

- `requestDeposit()` allows users to enter the depositQueue for their depositAsset by transferring the depositAmount to the SharesWrapper. This does not mint wrapped shares yet.
- `cancelRequestDeposit()` cancels the depositRequest and refunds the user.
- `depositFromQueue()` allows managers to process queued deposits, depositing to the vault and minting wrapped shares to the users. Users to be processed can be specified by address.
- `depositAllFromQueue()` allows managers to process all deposits that use the specified depositAsset.
- `setDepositMode()` allows managers to switch between the DepositModes Direct (deposit at any time) and Request.

There is a separate depositQueue for each depositAsset. Users can specify that they want to use native Ether, which the wrapper will automatically convert to and from WETH. There is no slippage protection on deposit and withdrawal requests. Finally, deposits are currently supported only for vaults deployed by Fund Deployer V4.

2.2.2 ParaswapV5

The ParaswapV5 adapter was extended to support the `MegaSwap` and `SimpleSwap` functionalities of the protocol. The incoming amounts from the trades are sent directly to the vaults. When using `takeMultipleOrders()`, unspent tokens are returned to the vault after the end of the trade. `takeOrder()` assumes that Paraswap does not leave any remainders of spent assets.

2.2.3 1inch swaps

The `1inchV5` adapter is introduced. Vaults can trade their assets by calling `swap` of `1inch`. The incoming amounts from the trades are sent directly to the vaults. The unspent tokens are also returned to the vault after the end of the trade.

2.2.4 ZeroExV4

The adapter for `ZeroExV4` is introduced. Enzyme vaults can submit orders as takers. Order makers can be restricted to only whitelisted ones by a list in the address list registry. The trades include only ERC20 tokens in the asset universe of Enzyme. The adapter supports the `fillOrKillLimitOrder` and `fillOrKillRfqOrder` calls. The interaction with ZeroExV4 is assumed to fully fill the taker's order and not leave a remainder of spent assets.

2.2.5 Roles and Trust Model

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

In general, we assume Enzyme only interacts with normal ERC-20 tokens that do not have multiple entry points, callbacks, fees-on-transfer, or other special behaviours.

Fund owners and asset managers are generally fully trusted for a fund. However, their powers can be limited through the fund's settings. The funds settings/policies are assumed to be set up correctly for the intended configuration/usage.

The vault owner can take control of any GatedSharesWrapper shares at any time. The wrapper managers can force wrapper redemptions or disallow redemptions at any time.

Governance is fully trusted and expected to not only behave honestly but also to fully understand the systems they are interacting with which includes choosing appropriate parameters.

All external systems are expected to be non-malicious and work correctly as documented.

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:

- Security: Related to vulnerabilities that could be exploited by malicious actors
- 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

5.1 No Asset Check in `requestDeposit()`

Design Low Version 1 Risk Accepted

CS-SUL10-001

The `requestDeposit` function in *GatedRedemptionQueueSharesWrapperLib* takes a `_depositAsset` argument. This asset should match up with one of the deposit assets of the corresponding vault. If they do not match, it will be impossible for the manager to call `__depositFromQueue` for that asset.

As there is no check on `_depositAsset`, a user may accidentally deposit an incorrect asset. A user accidentally making such a request can get their funds back by calling `cancelRequestDeposit`.

Note that in Enzyme V4 each vault only has a single deposit asset, but this may change in the future.

Risk Accepted:

Avantgarde Finance acknowledges that this can happen and accepts the risk that a user could waste gas.

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	1
Medium-Severity Findings	0
Low-Severity Findings	0

- Incorrect depositFromQueue Can Lead to Loss of User Funds Code Corrected

6.1 Incorrect `depositFromQueue` Can Lead to Loss of User Funds

Security High Version 1 Code Corrected

CS-SUL10-002

`depositFromQueue()` in GatedRedemptionShareWrapper takes an array of addresses and processes their deposit, removing them from the DepositQueue. It reads the user's position `index` in the queue from storage. Afterwards, the `Request` at `index` is deleted.

However, there is no check if the address actually had a `Request` in the queue to begin with. If a user has no request, their storage mapping will point to a `Request` with the default `index` of zero. This will result in the first `Request` in the queue being deleted. The user who had the deleted request will not receive any wrapped shares and their funds will be lost.

As `__depositFromQueue()` is protected with `__onlyManagerOrOwner()`, only the managers can call the function with a non-existing address, which would cause the issue.

This could happen on 2 occasions:

1. The manager accidentally passes an incorrect address.
2. The manager's `depositFromQueue` call gets frontrun by one of the users in the DepositQueue calling `cancelRequestDeposit()`. Now the canceled `Request` no longer exists, leading to a loss of funds for another user. If the manager passed all users in the queue as argument, the attacker will also need to deposit again from another address after they cancel, otherwise the queue will not have a sufficient length and will revert. Note that the attack is more likely to happen when there's no whitelisting in place i.e., when `useDepositApprovals` is set to false.

`depositAllFromQueue()` is not affected, as here incorrect addresses cannot be passed.

Code corrected:

The internal `__removeDepositRequest` function now validates that a `Request` has an `assetAmount` that is greater than zero. This ensures that the `Request` must exist before removing it. If a non-existant `Request` is passed, the function now reverts.

6.2 No minIncomingAsset Check for ZeroEx

Informational | Version 1 | Code Corrected

CS-SUL10-003

In ZeroExV4Adapter, the `parseAssetsForAction` function returns a `minIncomingAssetAmounts_` array with only a zero value. This means there is no internal check on the trade price, as is done in other adapters.

As only single orders with a fixed price can be taken on ZeroEx, there can be no slippage.

However, a vault admin may accidentally take an order with a lower price than they intended.

Code corrected:

The `parseAssetsForAction` function now sets the `minIncomingAssetAmounts_` to the amount that is expected to be received given the price of the order and the taker 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 Cannot Pay ZeroEx Protocol Fee

Note Version 1

The ZeroEx protocol has a fee mechanism, where a fee in native ETH must be attached to fill limit orders. The `ZeroExV4Adapter` does not support sending ETH with the limit order call to pay this fee.

However, the fee is currently set to zero as of the time of this report and has been since September 29, 2021.

If the fee is set to a non-zero value again in the future, the adapter will no longer be able to make Limit Order trades.

7.2 Some ZeroEx Order Restrictions Not Supported

Note Version 1

ZeroEx orders have `taker` and `txorgin` fields, which restrict who can take a particular order. These restrictions cannot be used to specify Enzyme vaults as multiple vaults can use the same ZeroExV4Adapter adapter.

If, for example, the `taker` field was used to restrict an order, it would be set to the address of the adapter. Any caller of the adapter would be able to take the order.