PUBLIC

Code Assessment

of the Sulu Extensions XVII Smart Contracts

May 21, 2024

Produced for

by

Contents

1 Executive Summary

Dear all,

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

Avantgarde Finance implements a new Enzyme external position to integrate with Alice, a managed gateway to facilitate orders of whitelisted accounts/smart contracts and an off-chain marketplace.

The most critical subjects covered in our audit are functional correctness, asset solvency and the integration in Sulu. 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	0
Medium-Severity Findings	0
Low-Severity Findings	1
• Code Corrected	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 Enzyme Protocol repository. No documentation, but access to the private code of Alice was provided at commit `6653f5c56670e1f0f4b9a2e9723fadefc4bf2bee`. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	7 May 2024	6d3b125c8d662d1c08e4b6964d6dd4531a0b2733	Initial Version
2	17 May 2024	40a6053527afbcac3ead4a0ffbdd1cb11f6ce444	Removed infinite approval

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

The scope for the Alice Protocol external position is:

```
contracts/external-interfaces/IAliceOrderManager.sol
contracts/release/extensions/external-position-manager/external-positions/alice/AlicePositionLib.sol
contracts/release/extensions/external-position-manager/external-positions/alice/AlicePositionParser.sol
contracts/release/extensions/external-position-manager/external-positions/alice/IAlicePosition.sol
contracts/release/extensions/external-position-manager/external-positions/alice/bases/AlicePositionLibBase1.sol
```

2.1.1 Excluded from scope

Any contracts inside the repository that are not mentioned in `Scope` are not part of this assessment. All external libraries, and imports are assumed to behave correctly according to their high-level specification, without unexpected side effects.

The correctness of external systems is not in scope.

Tests and deployment scripts are excluded from the scope.

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 a new external position for integrating with Alice.

2.2.1 External Position: Alice

Alice implements a manually controlled order manager. Only whitelisted participants (i.e., whitelisted external position proxy instances) can place orders to exchange assets. Supported asset pairs are defined in active `Instruments` of Alice.

1. `PlaceOrder`: Places an order (`ProjectAlice.placeOrder()`). The parameters passed specify the `instrument` (pair of assets), `isBuyOrder`, `quantityToSell` and `limitAmountToGet`. The funds are withdrawn from the vault and escrowed to the Alice contract. If the asset is the native token, the wrapped native token received from the vault is unwrapped and passed along as call value.
2. `Sweep`: Orders are fulfilled or cancelled by the operator of the `ProjectAlice` contract. This moves the funds to the external position. Action `Sweep` facilitates to move these assets to the vault. Based on the `orderIds` passed, the involved assets are retrieved. If these orders are settled or cancelled, non-zero balances of the involved assets are moved from the external position to the vault.
3. `RefundOrder`: After a certain timeout, if a previously placed order of the external position is not settled or is cancelled by the operator, this action allows to refund the order (`ProjectAlice.refundeOrder()`). Retrieved Funds are pushed to the vault, this includes the native asset which is automatically wrapped at the vault.

This position cannot have debt hence `getDebtAssets()` always returns zero.

`getManagedAssets()` evaluates the managed assets of the position. Managed assets include assets held at this contract, i.e., assets received after an order has been settled or cancelled as well as open orders where the funds are currently escrowed at the ProjectAlice contract. Based on the tracked orders, all the active orders are evaluated based on their order value. For tracked orders which are inactive the involved assets are checked for non-zero balance of the external position. The aggregated sum is the value of the external position.

Public getters `getOrderDetails()`, `getOrderIds()` are provided to retrieve the IDs of tracked orders and their details, respectively.

2.2.2 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 behaviors.

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.

Governance is fully trusted and expected to not only behave honestly but also to fully understand the systems Enzyme is interacting with.

External systems (notably Alice) are expected to work correctly and as expected. Furthermore, since Alice is a centralized system with powerful roles (managing the executions, ability to change the configuration), Alice must be fully trusted.

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

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	1

- Infinite Approvals Code Corrected

6.1 Infinite Approvals

Security Low Version 1 Code Corrected

CS-SUL17-002

When placing an order the external position grants `ProjectAlice` an unlimited allowance to transfer the ERC-20 token involved. The external position may hold non-zero balance of such tokens, e.g. due to orders having been settled or cancelled and hence tokens having been transferred to the external position. Although in the current state of the code there are no possible misuse cases, e.g. if future versions of Alice show a different behavior, it could be dangerous. Generally, minimizing risk by granting only the necessary approvals is best practice.

Code corrected:

Avantgarde Finance has limited the token approval to `placeOrderArgs.quantityToSell`:

```
IERC20(outgoingAssetAddress).safeApprove({
    _spender: address(ALICE_ORDER_MANAGER),
    _value: placeOrderArgs.quantityToSell
});
```

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 Sweep May Leave Empty Orders Tracked

Informational **Version 1**

CS-SUL17-001

`Actions.Sweep` removes the order being processed from the list of the tracked orders and retrieves any non-zero balance of the orders incoming and outgoing assets to the vault. These balances may include funds of other settled or cancelled orders which may not be included in the list of orders to be processed. Such orders stay tracked, even though their balances have already been moved to the vault. This has no consequences, except for higher gas consumption during fund evaluation. To remove these orders eventually, action sweep must be executed for their respective `orderId`.

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 Place Order: `limitAmountToGet` Not Guaranteed

Note Version 1

Action `PlaceOrder` / the arguments for `ProjectAlice.placeOrder()` include a parameter `limitAmountToGet`. This must not be misunderstood as the guaranteed minimum amount out of the order.

When the order is settled, Alice only enforces that the amount out before the fee is taken (`_quantityReceivedPreFee` passed as function argument) exceeds `limitAmountToGet`. This check is not done on the actual amount of tokens transferred to the external position.

Malicious actions may result in funds lost:

- The actual amount out the external position receives sees the fee deducted. The fee, a percentage of the amount, can be updated anytime by Alice contract admin, but it cannot exceed the `maxFeeRate` set at deployment, which is unchangeable. `maxFeeRate` is only guaranteed to not exceed 100%. Users should be aware of the `maxFeeRate` actually set and take this into consideration.
- More directly, the escrowed funds may be drained e.g., by replacing the liquidityPoolContract with a modified version allowing to seize the funds while `ProjectAlice.settle()` executes. `limitAmountToGet` has no effect in such a scenario.

As defined in the trust model, Alice must be fully trusted.