PUBLIC

Code Assessment

of the Sulu Extensions XXIV Smart Contracts

May 26, 2025

Produced for

by

Contents

1 Executive Summary

Dear all,

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

Enzyme Foundation implements an upgrade for the Stakewise v3 external position to add support for newer versions of Stakewise v3 vaults while disabling the support for versions supported previously. Note that the upgrade is required due to breaking changes in Stakewise v3 that were not covered in the initial review of the Stakewise v3 external position.

The most critical subjects covered in our audit are functional correctness, correct integration with the external system, and front-running. The general subjects covered are gas efficiency and access control.

Security regarding all the aforementioned subjects is high. The most notable issue Unpriced left tickets, resulting in computing incorrect position values, has been resolved.

In summary, we find that the codebase provides a good 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 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	08 May 2025	1080e9554108a5e8188fa8f0fd5dae806592c786	Initial Version
2	20 May 2025	e70c26efb5f8c6139cc7b0e913c22dc6750abaa7	After Intermediate Report

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

The following files were in scope:

```
contracts/
    release/extensions/external-position-manager/external-positions/stakewise-v3-staking/
        bases/StakeWiseV3StakingPositionLibBase1.sol
        IStakeWiseV3StakingPosition.sol
        StakeWiseV3StakingPositionDataDecoder.sol
        StakeWiseV3StakingPositionLib.sol
        StakeWiseV3StakingPositionParser.sol
    external-interfaces/
        IStakeWiseV3EthVault.sol
        IStakeWiseV3VaultsRegistry.so
```

2.1.1 Excluded from scope

All files not explicitly mentioned above are out of scope. For StakewiseV3, we assume that only implementations for EthVault and EthGenesisVault at tags `v3.0.0` and `v3.0.1` are used for the external position. Further, we expect that prior to the upgrade, no pending exit requests exist.

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.

Stakewise v3 is a liquid staking protocol where users can create vaults for their liquid staking derivative. Enzyme Foundation implements an external position for StakewiseV3 vaults. More specifically, an external position compatible with EthVault and EthGenesisVault at tags `v3.0.0` and `v3.0.1` is provided. Vault tokens could be unified as osETH via a lending mechanism using them as collateral, however note osETH integration is not supported by the External Position.

The external position implements the following actions:

- `Stake`: Stake to a given vault and receive vault tokens. Note that it is not required to deposit 32 ETH (or a multiple of it).
- `EnterExitQueue`: Signal an exit. Eventually, the funds will be unstaked. Note that vault tokens are burned immediately and a position ticket representing the exit is generated. Further, in some corner cases instant withdrawals can be possible.
- `ClaimExitedAssets`: Claim the exited assets. Potentially, not all assets will be claimed and yet another position ticket will be received for the assets not exited yet.

Note that an action `Redeem` exists but reverts as it is a legacy action.

`getManagedAssets()` returns the value of vault shares plus the value of the exit tickets. Note that the value can be computed by leveraging `convertToAssets()` (value for shares or shares not possible to exit) and `calculateExitedAssets()` (value of shares possible to exit). Since no debt is taken, `getDebtAssets()` return empty arrays.

Please see Share Price Arbitrage and Avoiding of Slashing for further consideration.

2.3 Trust Model

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

Asset managers are trusted to not deposit into malicious vaults that purposefully get slashed as this could lead to loss of user funds.

While the implemenation must be whitelisted, Stakewise is trusted to provide only vaults that return the actual implementation of vaults. Further, Stakewise is trusted to not allow any stealing of funds through upgrades.

Governance is trusted to perform the upgrade of the position according to Assessment Overview.

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 Open 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:

- Correctness: Mismatches between specification and implementation

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 Open 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

- Unpriced Left Tickets Code Corrected

6.1 Unpriced Left Tickets

Correctness Low Version 1 Code Corrected

CS-SUL-24-001

Note that in Stakewise V3 withdrawal requests could be partially withdrawable. Even though such scenarios are seemingly uncommon, the value of a request is constituted by the withdrawable amount and the value of the pending tickets. However, the pricing of exit requests computes the value of exit requests as follows:

```
(,, uint256 claimedAssets) = stakeWiseVault.calculateExitedAssets({
    _receiver: address(this),
    _positionTicket: exitRequest.positionTicket,
    _timestamp: exitRequest.timestamp,
    _exitQueueIndex: uint256(exitQueueIndex)
});
```

Ultimately, unclaimable exit shares might be left (`leftTickets`) and could be left unpriced in `getManagedAssets()`. As a consequence, the position could remain undervalued.

Code corrected:

The code has been adjusted to account for the left tickets. More specifically, if tickets would be left, they are treated at a 1:1 ratio with shares as they are not redeemable yet.

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 Share Price Arbitrage and Avoiding of Slashing

Note Version 1

Note that users could avoid slashing by exiting their shares for the underlying vault assets prior to the slash being realized on-chain. As a consequence, other users could be slashed more.

Similarly, the share price could be undervalued if rewards are to be expected but are not realized yet.

7.2 StakeWise Deposit May Revert

Note Version 1

Stakewise v3 deposits can revert if harvesting of rewards is required by the vault. In such cases, `deposit()` could revert. As an alternative, `updateStateAndDeposit()` could be used. However, since the external position does not implement such an action, the asset manager could be required to call `updateState()` manually before depositing.