decurity

Smart Contract Security Audit Report

Gearbox Integrations Audit

1. Contents

2. General Information

This report contains information about the results of the security audit of the Gearbox (hereafter referred to as “Customer”) smart contracts, conducted by Decurity in the period from 2025-12-24 to 2025-12-30.

2.1. Introduction

Tasks solved during the work are:

- Review the protocol design and the usage of 3rd party dependencies,
- Audit the contracts implementation,
- Develop the recommendations and suggestions to improve the security of the contracts.

2.2. Scope of Work

The audit scope included the contracts in the following repository: Gearbox-protocol/integrations-v3/. Initial review was done for the commit 9d3c72a06d82c52418fa86b8d4a8385052af7727 and the re-testing was done for the commit 047163d347febcfdd09a609edfe192355a6ba529.

The following contracts have been tested:

- contracts/adapters/kelp/KelpLRTDepositPoolAdapter.sol
- contracts/adapters/kelp/KelpLRTWithdrawalManagerAdapter.sol
- contracts/helpers/kelp/KelpLRTDepositPoolGateway.sol
- contracts/helpers/kelp/KelpLRTWithdrawalManagerGateway.sol
- contracts/helpers/kelp/KelpLRTWithdrawalPhantomToken.sol
- contracts/helpers/kelp/KelpLRTWithdrawer.sol

2.3. Threat Model

The assessment presumes actions of an intruder who might have capabilities of any role (an external user, token owner, token service owner, a contract). The centralization risks have not been considered upon the request of the Customer.

The main possible threat actors are:

- User,
- Protocol owner,
- Liquidity Token owner/contract.

2.4. Weakness Scoring

An expert evaluation scores the findings in this report, an impact of each vulnerability is calculated based on its ease of exploitation (based on the industry practice and our experience) and severity (for the considered threats).

2.5. Disclaimer

Due to the intrinsic nature of the software and vulnerabilities and the changing threat landscape, it cannot be generally guaranteed that a certain security property of a program holds.

Therefore, this report is provided "as is" and is not a guarantee that the analyzed system does not contain any other security weaknesses or vulnerabilities. Furthermore, this report is not an endorsement of the Customer's project, nor is it an investment advice.

That being said, Decurity exercises best effort to perform their contractual obligations and follow the industry methodologies to discover as many weaknesses as possible and maximize the audit coverage using the limited resources.

3. Summary

As a result of this work, we have not discovered any exploitable security issues.

The other suggestions included fixing the low-risk issues and some best practices.

The team has given the feedback for the suggested changes and explanation for the underlying code.

3.1. Suggestions

The table below contains the discovered issues, their risk level, and their status as of January 12, 2026.

Table. Discovered weaknesses

Issue	Contract	Risk Level	Status
stETH rounding issues may break completeWithdrawal	contracts/helpers/kelp/KelpLRTWithdrawer.sol	**Low**	Fixed
Unnecessary Ownable inheritance	contracts/helpers/kelp/KelpLRTWithdrawalPhantomToken.sol	**Info**	Fixed

4. General Recommendations

This section contains general recommendations on how to improve overall security level.

The Findings section contains technical recommendations for each discovered issue.

4.1. Security Process Improvement

The following is a brief long-term action plan to mitigate further weaknesses and bring the product security to a higher level:

- Keep the whitepaper and documentation updated to make it consistent with the implementation and the intended use cases of the system,
- Perform regular audits for all the new contracts and updates,
- Ensure the secure off-chain storage and processing of the credentials (e.g. the privileged private keys),
- Launch a public bug bounty campaign for the contracts.

5. Findings

5.1. stETH rounding issues may break completeWithdrawal

Risk Level: **Low**

Status: Fixed in the commit.

Contracts:

- contracts/helpers/kelp/KelpLRTWithdrawer.sol

Description:

`completeWithdrawal` uses a hardcoded tolerance of 10 wei:

```
if (onWithdrawer < amount && onWithdrawer > amount - 10) amount =
onWithdrawer;
```

The current maximum rounding error is capped at 2 wei. With the current numClaimableRequests maximum of 5, the accumulated error stays within this bound, so the function behaves as expected. However, the maximum error may increase in the future to 3 wei or more, which increases the risk that the hardcoded tolerance becomes insufficient or inconsistent with the intended rounding/error model.

Remediation:

Consider increasing or parameterizing the tolerance to account for potential future increases in the rounding error.

References:

- https://github.com/lidofinance/core/issues/442

5.2. Unnecessary Ownable inheritance

Risk Level: **Info**

Status: Fixed in the commit.

Contracts:

- contracts/helpers/kelp/KelpLRTWithdrawalPhantomToken.sol

Description:

KelpLRTWithdrawalPhantomToken inherits from Ownable but doesn't use utilise owner-only access control.

Remediation:

Consider removing the Ownable inheritance from KelpLRTWithdrawalPhantomToken if no owner-gated logic is required.

6. Appendix

6.1. About us

The Decurity team consists of experienced hackers who have been doing application security assessments and penetration testing for over a decade.

During the recent years, we've gained expertise in the blockchain field and have conducted numerous audits for both centralized and decentralized projects: exchanges, protocols, and blockchain nodes.

Our efforts have helped to protect hundreds of millions of dollars and make web3 a safer place.