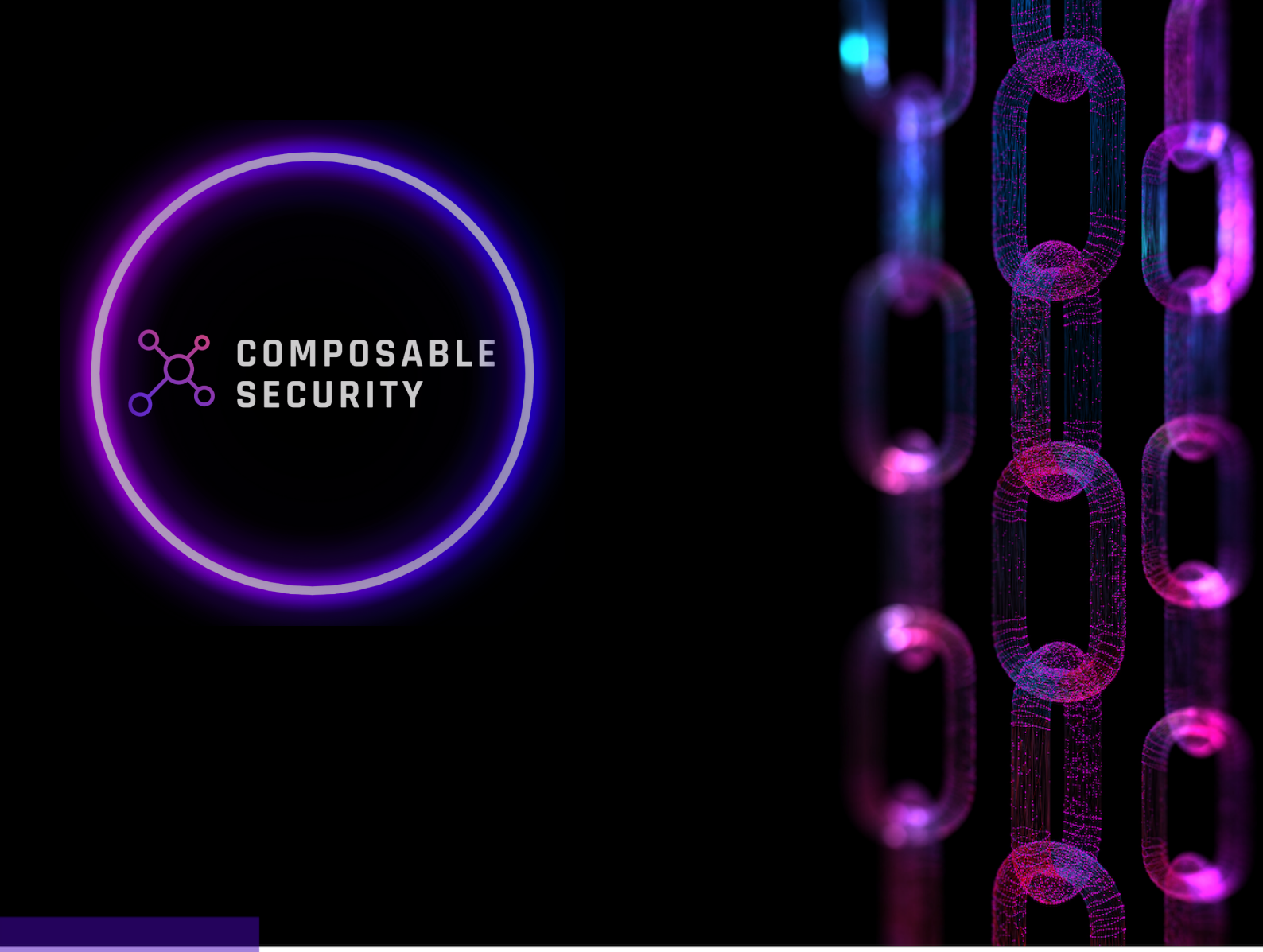

COMPOSABLE SECURITY

# REPORT

## Security consultation for Lido

Prepared by: Composable Security
Report ID: LDO-f17f089
Test time period: 2025-08-20 - 2025-08-22
Report date: 2025-08-22
Version: 1.0
Visit: composable-security.com

# Contents

# 1. Security consultation summary (2025-08-22)

## 1.1. Subject of consultation

The **Composable Security** team was commissioned by the **Lido** to conduct a security review of fix for Issue 740 and changes related to previous workarounds. The team focused on auditing the changes from 5.1.0 (already audited) → 5.4.1, to skip all the removed items.

This upgrade addresses a vulnerability, that introduces possibility to influence the report (and rewards distribution) by uploading the validator keys already existing on CL.

The proposed solution adds filtering by `used=true` when fetching data from KAPI for reward distribution calculation in the `get_module_validators_by_node_operators method` to exclude not used keys.

The Composable Security team spent 3 days identifying and covering the following scenarios:

- Uploading the validator keys already existing on CL
- Injecting attacker-chosen (pre-computed) bytes32 values into Consensus-Layer blocks to present them as the root of a historical block.
- Front-running NodeRegistry to redirect reward distribution to an emergency vault.
- Mismatched/stale KAPI response.

The detailed description of all covered scenarios can be found in 2.

## 1.2. Consultation results

The primary risk associated with this upgrade is the potential for influencing the report and rewards distribution.

After conducting a consultation, the security assessment did not identify any vulnerabilities introduced by this fix that could directly compromise the security or operational integrity of the Oracle system.

It is recommended to perform the following action:

- Audit KAPI service to verify the statement regarding the `used` parameter, especially to verify the following scenarios:
  - Adding a key that is already used by other Node Operator.
  - Adding the same used key again (even if unvetted it will be returned by KAPI, potentially as used).

Following the implementation of these actions, the upgrade is considered secure under the following assumption: **when the `used` parameter is set, it is guaranteed that the key was deposited initially through the DSM module and is active under the control of the Node Operator.**

## 1.3. Scope

The subjects of the test were changes from 5.1.0 (already audited) → 5.4.1.

**GitHub repository:**
https://github.com/lidofinance/lido-oracle
**CommitID:** f17f0898cd8c46eefba5da0ad3162dc2f4bcf439

**Documentation:**

- Lido Docs

## 1.4. Deployments

After the security review conducted, we verified that the Dockerfile used to build an image uses the reviewed source code. The published image with the manifest digest **sha256:db0d00468df9840aa4084485314911a030c39c57da80656e92152883b2da6566** corresponds to the commitID: **f17f0898cd8c46eefba5da0ad3162dc2f4bcf439**.

## 1.5. Disclaimer

Security consultation **IS NOT A SECURITY WARRANTY**.

During the review, the Composable Security team makes every effort to detect any occurring problems and help to address them. However, it is not allowed to treat the report as a security certificate and assume that the project does not contain any vulnerabilities. Securing applications is a multi-stage process, starting from threat modeling, through development based on best practices, security reviews and formal verification, ending with constant monitoring and incident response.

***Therefore, we encourage the implementation of security mechanisms at all stages of development and maintenance.***

# 2. Covered scenarios

## 2.1. Uploading the validator keys already existing on CL

Prior to the fix, a Node Operator could upload a validator pubkey that was already active on the consensus layer but never deposited via the module. The attribution logic in `get_module_validators_by_node_operators` would still match that CL validator to the operator, because it builds its map from all module keys (as stated in the swagger documentation "Filter to get used keys. Possible values: true/false. If this value is not specified, endpoint will return all keys."). This resulted in misallocation of rewards to validators that were not actually funded by the module.

The `MODULE_OPERATORS_KEYS` was converted to `USED_MODULE_OPERATORS_KEYS` and specifies the parameter `used=true`. As a consequence, uploading a pubkey that already exists in CL no longer affects attribution unless and until the deposit is performed through the module and becomes visible at the chosen `blockstamp`.

The **Lido** team stated that the `used` parameter is defined per staking module and per key. In addition, when the parameter is set, it is guaranteed that the key was deposited initially through the DSM module and is active under the control of the Node Operator.

## 2.2. Injecting attacker-chosen (pre-computed) bytes32 values into Consensus-Layer blocks to present them as the root of a historical block.

A lack of input validation in the `processHistoricalWithdrawalProof` method allowed for the submission of fake block hashes.

Immediate mitigation involved an off-chain fix to prevent the burning of Node Operator stETH bonds.

The interim off-chain changes were safely undone, and scenario is fully mitigated by an on-chain upgrade. It removed the vulnerable method `processHistoricalWithdrawalProof` from the `CSVerifier`.

The planned for September permanent fix will re-enable the method but calculate the historical block GIndex on-chain instead of accepting user input. The updated `CSVerifier` will also introduce a `GateSeal` for added security, allowing it to be paused.

## 2.3. Front-running NodeRegistry to redirect reward distribution to an emergency vault

Initially, a vulnerability existed due to a legacy Aragon function, `transferToVault`. This function could be exploited to redirect undistributed node operator rewards from the `NodeOperatorsRegistry` contract to the DAO Agent.

As an immediate mitigation, an off-chain hotfix was applied. This solution bundled the report data submission with the reward distribution, executing them through a private mempool to prevent the exploit.

For a permanent resolution, a definitive on-chain fix was subsequently deployed. This fix disables the legacy function by setting the `RecoveryVaultAppId` to the null value, causing any attempt to call `transferToVault` to fail. With this permanent solution in place, the temporary off-chain hotfix was removed. The scenario is fully mitigated.

## 2.4. Mismatched/stale KAPI response

The scenario where responses from KAPI could be stale or poisoned was verified to make sure that the Oracle code correctly verifies returned data. Both crucial parameters block timestamp and module address are verified to make sure that the response is valid.


## Damian Rusinek

Smart Contracts Auditor

@drdr_zz

damian.rusinek@composable-security.com



## Paweł Kuryłowicz

Smart Contracts Auditor

@wh01s7

pawel.kurylowicz@composable-security.com