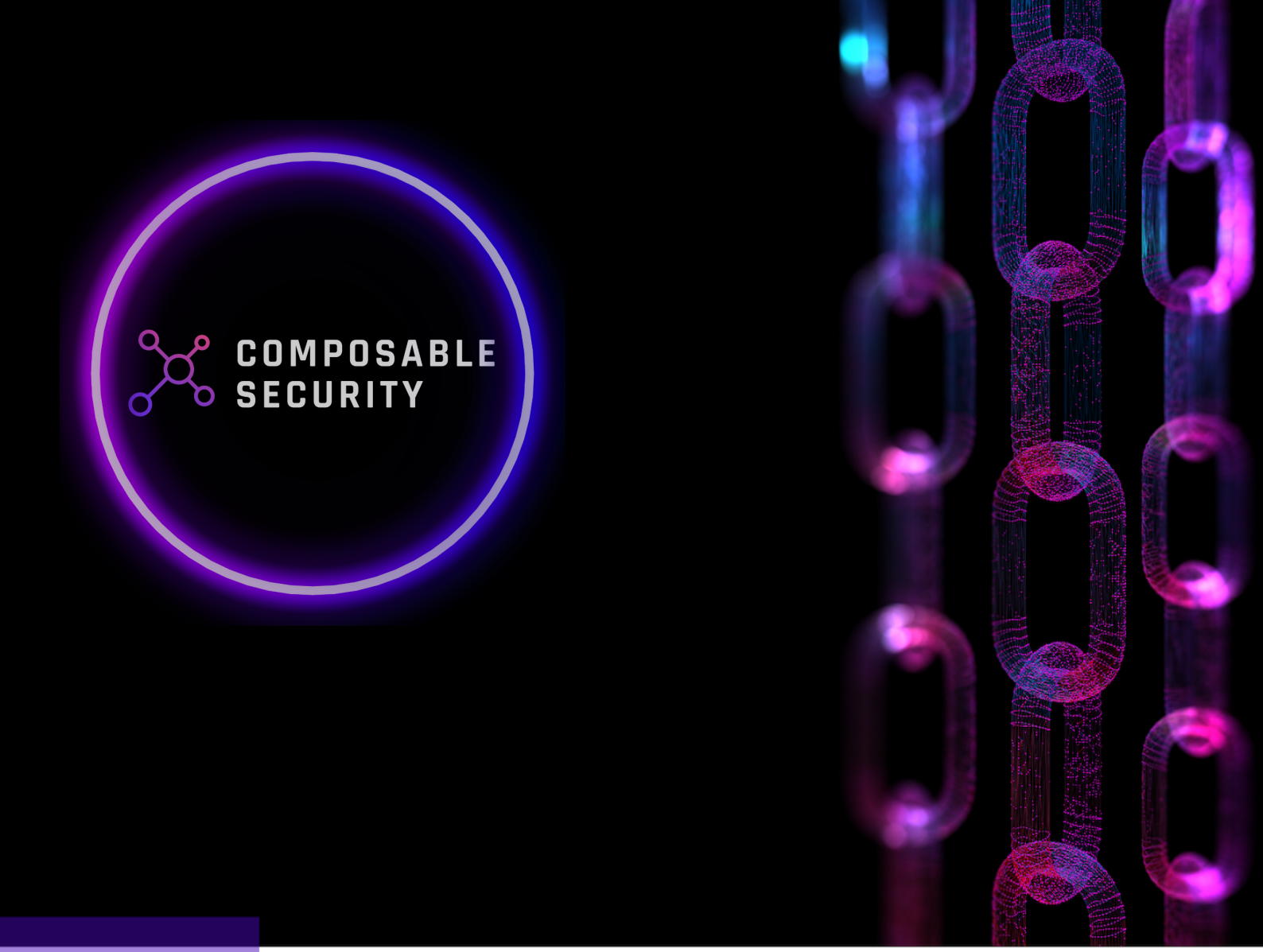

COMPOSABLE SECURITY

REPORT

Security consultation for Lido

Prepared by: Composable Security
Report ID: LDO-05b54cc9
Test time period: 2025-06-20 - 2025-06-21
Retest time period: 2025-06-23 - 2025-06-23
Report date: 2025-06-24
Version: 1.0
Visit: composable-security.com

Contents

1. Retest summary (2025-06-23)

1.1. Results

The **Composable Security** team participated in a one-time iteration to verify if the consultation recommendations were properly implemented.

The recommendations to ensure the upgrade is deployed to all Oracle members who may act as submitters prior to the on-chain fix deployment and to properly configure those nodes cannot be checked in the code, but the Lido team is aware of those and plan to provide all information to node operators.

The default value for `PRIORITY_FEE_PERCENTILE` used to calculate the maximum priority fee is still 3, but the team will inform node operators to set the ENV variable to 95.

The team decided to not make the relayers list mandatory for three reasons:

- In case there are some issues with the infrastructure that require additional work to build the oracle image and deploy on testnet (because it is done automatically with each release),
- It might be useful to have a way to turn off this feature after the fix is delivered on-chain.
- This ENV variable might not be applicable properly to testnets.

The number of bundles sent to relayers has been changed to 6 blocks.

Additionally, the bundle timeout has been increased to 15 minutes after finalization of the slot, becuase the average second phase report delivery time is 12-13 minutes after finalization.

1.2. Scope

The retest scope included the changes, on a different commit in the same repository.

GitHub repository: https://github.com/lidofinance/lido-oracle
CommitID: 701306dfba316ea4b8b3c0714a7297dc9ce6594b

2. Security consultation summary (2025-06-21)

2.1. Subject of consultation

The **Composable Security** team was commissioned by the **Lido** to conduct a security review of a proposed hotfix for the Lido Oracle.

This upgrade addresses a vulnerability, identified by a third-party, that enables a front-running attack on the `NodeRegistry` smart contract. The vulnerability could allow an attacker to redirect reward distributions to an emergency vault, necessitating a manual recovery and redistribution process by the Lido DAO.

The proposed interim solution involves atomicity through bundling the report data submission and reward distribution, which is to be executed via a private mempool to prevent front-running.

The Composable Security team spent 2 days to identify and cover the following scenarios:

- Submission by a node that does not have the fix update
- Missing report due to delayed execution of the transaction bundle
- Insufficient fees for the relayers
- Bundling distribution of rewards only for one module
- Resending the same extra data

The detailed description of all covered scenarios can be found in 3.

2.2. Consultation results

The primary risk associated with this upgrade is the potential for denial of service scenarios that could render the Oracle non-functional following deployment.

After conducting a consultation, the security assessment did not identify any vulnerabilities introduced by this hotfix that could directly compromise the security or operational integrity of the Oracle system. However, caution should be exercised and additional safeguards should be taken to minimize the risk.

Recommendations:

It is recommended to implement the following mitigation measures:

- Ensure the upgrade is deployed to all Oracle members who may act as submitters prior to the on-chain fix deployment.
- Update the configuration for all upgraded nodes (e.g., specify the relayer list as a mandatory requirement).
- Recalculate bundle parameters for a reduced block range (6 blocks) and increase the maximum priority fee.

- Increase the maximum priority fee to the highest value from the last block.

Following the implementation of these measures, the upgrade is considered secure under the following assumption: **In the event of extended bundle execution delays (exceeding 20 minutes), Oracle will revert to the previous vulnerable execution flow.**

2.3. Scope

The subjects of the test were changes in the selected Pull Request from the **Lido** repository.

GitHub repository:
https://github.com/lidofinance/lido-oracle
CommitID: 05b54cc968ba3301c6b1237cba4c08ca1e21c893

Pull Request: #719

Documentation:

- Lido Docs

2.4. 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 smart contract platforms 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.

3. Covered scenarios

3.1. Submission by a node that does not have the fix update

There are so-called fast lane members in the Oracle committee, which is a subset of members - different for each frame - that can exclusively submit the report data for approximately 640 seconds since the finalization of frame's reference slot.

It's important to upgrade the nodes of all members that will be fast lane members for all frames before the on-chain fix is deployed. Additionally, make sure all of them are fully configured, namely have the list of relayers specified (which should be required).

There exists a role that has no delay for submitting report data - `SUBMIT_DATA_ROLE`. However, currently no address has this role.

3.2. Missing report due to delayed execution of the transaction bundle

The execution of bundle transaction highly depends on the private pool relayers and can be delayed for a long time when the bundle gets rejected. After the delay reaches the length of the frame, this can lead to lack of processing of the report for a particular frame.

However, the upgrade handles the case when the bundle is not processed for longer than 15 minutes since the time when the frame's reference slot is finalized. Then, the Oracle falls back to the default execution that is again vulnerable to the reported issue.

This is a best-effort approach that balances the risks between the impact of the reported issue and the likelihood of long-term delay of bundle execution.

3.3. Insufficient fees for the relayers

Bundle execution is contingent upon the maximum priority fees being sufficiently high to incentivize relayers to select the bundle. Additionally, if the maximum fee cannot cover the base fee and priority fee, the transaction will be rejected.

Currently, the base fee is calculated as twice the base fee from the previous block, and the maximum priority fee is set to the median of maximum priority fees from the previous block. The upgrade submits the same bundle for the next 32 blocks to increase the probability of inclusion in upcoming blocks.

It is recommended to increase the percentile of maximum priority fees to select the highest maximum priority fee from the previous block.

Regarding the base fee calculation, the same fees are used to submit bundles for the next 32 blocks. In scenarios where transaction volume increases significantly (resulting in higher gas consumption), the base fee can increase by 12.5% over the next 6 blocks, potentially rendering the maximum fee insufficient to cover the required gas costs.

It is recommended to modify the bundle submission flow. The current approach is:

1. Calculation of parameters (including fees)
2. Submission of the bundle for 32 blocks
3. Wait for 32 blocks

A more robust approach would be:

1. Calculation of parameters (including fees)
2. Submission of the bundle for 6 blocks
3. Wait for 6 blocks
4. If the bundle is not included in any of the 6 blocks, return to step 1 (maximum 4 iterations).

3.4. Bundling distribution of rewards only for one module

The reported issue is related to all staking modules that require manual call to distribute rewards among Node Operators (through the `distributeReward` function). This includes Curated Staking and Simple SVT modules, but not the Community Staking Module. Those modules can be identified by the type value which is `curated-onchain-v1`.

The upgrade correctly includes all required modules by checking the type of each module.

3.5. Resending the same extra data

The `execute_module` function first processes the report data and then calls `process_extra_data` to handle additional data.

Following the upgrade, the extra data is included in the bundle, eliminating the need to call `process_extra_data` in this context. However, the `process_extra_data` function includes a check at the beginning to verify whether the data has already been submitted, and performs no action if it has been previously processed.

Furthermore, the `process_extra_data` function call remains necessary in scenarios where the system falls back to the previous execution flow.

Damian Rusinek

Smart Contracts Auditor

@drdr_zz

damian.rusinek@composable-security.com

Paweł Kuryłowicz

Smart Contracts Auditor

@wh01s7

pawel.kurylowicz@composable-security.com