LIDO ON POLYGON

0XORIO

MARCH 24, 2023

CONTENTS

1 INTRO

1.1 DISCLAIMER

The audit makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about the fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

1.2 ABOUT OXORIO

Oxorio is a young but rapidly growing audit and consulting company in the field of the blockchain industry, providing consulting and security audits for organizations from all over the world. Oxorio has participated in multiple blockchain projects during which smart contract systems were designed and deployed by the company.

Oxorio is the creator, maintainer, and major contributor of several blockchain projects and employs more than 5 blockchain specialists to analyze and develop smart contracts.

Our contacts:

- oxor.io
- ping@oxor.io
- Github
- Linkedin
- Twitter

1.3 SECURITY ASSESSMENT METHODOLOGY

A group of auditors is involved in the work on this audit. Each of them checks the provided source code independently of each other in accordance with the security assessment methodology described below:

1. Project architecture review

Study the source code manually to find errors and bugs.

2. Check the code for known vulnerabilities from the list

Conduct a verification process of the code against the constantly updated list of already known vulnerabilities maintained by the company.

3. Architecture and structure check of the security model

Study the project documentation and its comparison against the code including the study of the comments and other technical papers.

4. Result's cross-check by different auditors

Normally the research of the project is done by more than two auditors. This is followed by a step of mutual cross-check process of the audit results between different task performers.

5. Report consolidation

Consolidation of the audited report from multiple auditors.

6. Reaudit of new editions

After the provided review and fixes from the client, the found issues are being double-checked. The results are provided in the new version of the audit.

7. Final audit report publication

The final audit version is provided to the client and also published on the official website of the company.

1.4 FINDINGS CLASSIFICATION

1.4.1 Severity Level Reference

The following severity levels were assigned to the issues described in the report:

- **CRITICAL**: A bug leading to assets theft, locked fund access, or any other loss of funds due to transfer to unauthorized parties.
- **MAJOR**: A bug that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
- **WARNING**: A bug that can break the intended contract logic or expose it to DDoS attacks.
- **INFO**: Minor issue or recommendation reported to / acknowledged by the client's team.

1.4.2 Status Level Reference

Based on the feedback received from the client's team regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

- **NEW**: Waiting for the project team's feedback.
- **FIXED**: Recommended fixes have been applied to the project code and the identified issue no longer affects the project's security.
- **ACKNOWLEDGED**: The project team is aware of this finding. Recommended fixes for this finding are planned to be made. This finding does not affect the overall security of the project.
- **NO ISSUE**: Finding does not affect the overall security of the project and does not violate the logic of its work.
- **DISMISSED**: The issue or recommendation was dismissed by the client.

1.5 PROJECT OVERVIEW

Lido on Polygon is a liquid staking solution for MATIC backed by industry-leading staking providers. Lido lets users earn MATIC staking rewards without needing to maintain infrastructure and enables them to trade staked positions, as well as participate in on-chain decentralized finance with their staked assets.

1.6 AUDIT SCOPE

The scope of the audit includes changes made in PR#13 and PR#14 to the following contracts:

- PR#13 - StMATIC.sol
- PR#14 - StMATIC.sol

2 FINDINGS REPORT

OXORIO

2.1 CRITICAL

2.1.1 Incorrect recovery of `totalBuffered` parameter in `StMatic`

SEVERITY	CRITICAL
STATUS	NO_ISSUE

Description

In the function `recover` of the `StMatic` contract there is incorrect recovery of the `totalBuffered` parameter.
According to the report recovery plan is as follows:

> ...
> Add a function to `stMATIC` contract to fix the users affected by this bug:
> 1. Increase the `stMATIC` balance of all the users who staked after the bug (`+0.12%`).
> 2. Transfer the `102,470` remaining `MATIC` to the user.
> 3. Transfer the lost MATIC to the `stMATIC` contract through a direct transfer to cover the listed MATICs `1,309.425 MATIC`.
> ...

However, the current implementation does not take into account the lost `MATIC` on the `totalBuffered` balance.

```
// transfer the compensated amount to the affected user.
IERC20Upgradeable(token).safeTransfer(_compensatedAddress, _compensatedAmount);
totalBuffered -= _compensatedAmount;
```

At the same time, in the simulation, in order to get around this error, another recovery scenario is proposed.

```
const lostAmount = ethers.utils.parseUnits("1309.42584359816", 18);
const compensateAmount = ethers.utils.parseUnits("102470.7609", 18);
const compensateAddress = DAOSigner.address;
const compensateAddressBalanceBeforeRecover = await MATIC.balanceOf(
```

```
  compensateAddress
);

 // Exec the recover function
    await stMATIC
      .connect(DAOSigner)
      .recover(
        userAddresses,
        userBalances,
        compensateAddress,
        compensateAmount.sub(lostAmount)
    );

// Check if the totalBuffered stMatic is correct
    expect(totalPooledMaticAfter, "totalBuffered").eq(
      totalPooledBefore.sub(compensateAmount.sub(lostAmount))
    );

// transfer lost tokens to the stMatic contract (We have already transferred the half from the
recover).
await MATIC.connect(MATIC_WHALE).transfer(compensateAddress, lostAmount);

// Check if the compensateAddress received the compensateAmount.
const compensateAddressBalanceAfterRecover = await MATIC.balanceOf(
  compensateAddress
);

expect(
    compensateAddressBalanceAfterRecover,
    "compensateAddressBalanceAfterRecover"
).eq(compensateAddressBalanceBeforeRecover.add(compensateAmount));
```

The simulation is written incorrectly, and in order for the `totalBuffered` balance to be valid, the lost `MATIC` is subtracted from the amount of `compensateAmount`. And the lost `MATIC` amount is sent directly to the `compensateAddress`.

Thus, restoring according to the plan will lead to an incorrect value of `totalBuffered` and as a result, when redeeming `stMATIC`, users will receive less `MATIC` than expected.

Recommendation

We recommend revising the contract recovery plan and adding a separate feature to recover lost `MATIC` instead of direct transfer.

Update

Shard Labs:

The document was outdated.

For security, the amount transferred from stMATIC will go to LIDO multisig and then from it to the user in 1 transaction.
LIDO multisig is just a regular gnosis safe multisig currently used as a "DAO" role in Lido on Polygon that will receive those MATIC tokens and manually send them to the affected address.

2.2 MAJOR

No issues found

2.3 WARNING

No issues found

2.4 INFO

No issues found

3 CONCLUSION

Smart contracts have been audited and no issues were found.

THANK YOU FOR CHOOSING

OXORIO