# Lido for Polygon Smart Contracts Security Audit Report for PR#67

April 8, 2022

[TOC]

## Introduction

### Project overview

Lido on Polygon is a liquid staking solution for MATIC.

## Scope of the Audit

The scope of the audit includes checking the following fixes in PR#67:

- Added the StMatic NFT amount to the total Pooled Matic.
- Added function calculatePendingBufferedTokens to calculate the total amount hold
- Fixed getMaticFromTokenId to return the request from stMatic contract

The audited commit identifier is b992dc60ccb638227022e912fab3face9097fc41

## Findings Severity breakdown

### Classification of Issues

Every issue in this report was assigned a severity level from the following:

- CRITICAL: Bug leading to assets theft, fund access locking, or any other loss funds to be transferred to any party.
- MAJOR: Bug that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
- WARNING: Bug that can break the intended contract logic or expose it to DoS attacks.
- INFO: Other issue and recommendation reported to/ acknowledged by the team.

### Findings' breakdown status

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

- FIXED: Recommended fixes have been made to the project code and no longer affect its security.
- ACKNOWLEDGED: The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
- NO ISSUE: Finding does not affect the overall safety of the project and does not violate the logic of its work
- DISMISSED: This indicates that the issue was dismissed by the client.
- NEW: Waiting for project team's feedback on the finding discovered

## Report

### CRITICAL

No issues found

### MAJOR

#### [NEW] An attacker may make `submit` and `requestWithdraw` fail with out-of-gas

##### Description

Steps:

1. An attacker deposits some matic
2. He/she calls requestWithdraw(1) several times and receives NFTs. It costs him/her ~600k gas for each call.
3. Sends those NFTs to StMatic.
4. Each call to getMaticFromTokenId costs ~12k gas.
5. claimTokens2StMatic will not help until stakeManager.epoch() >= lidoRequests.requestEpoch, then the attacker may repeat the attack.
    - Epoch is ~3 hours, withdrawalDelay is 80 epochs (see block explorer) therefore withdrawalDelay is ~240 hours or ~10 days.
    - This is in contradiction with the Polygon doc which says that the unbonding period is ~3 hours * 80 = 3-4 days.
    - But even so, that is quite a lot of time.

Results:

Function calls `calculatePendingBufferedTokens` => `getTotalPooledMatic` => `convertStMaticToMatic` / `convertMaticToStMatic` => `requestWithdraw` and `submit` will be locked.
Users will not be able to neither deposit nor submit.

The attack costs 1800kk gas for 3000 NFTs, on a good day and time it could be ~36eth. It will increase the gas price of the `requestWithdraw` and `submit` calls up to 36kk, more than average block gas limit. But even a much smaller amount would make the price of participation too high so that almost no one would have any incentive to do it.

Also related, if there are a lot of withdrawal using `withdrawTotalDelegated` one day this call will start to fail with out-of-gas. Before that every call to `requestWithdraw` and `submit` will cast more and more. If there are 100 000 elements in `getOwnedTokens()` cost will be ~ 300 000 000 gas.

#### Recommendation

We recommend removing token from the owner2Tokens[anAddress] array on burn.

## WARNING

No issues found.

## INFO

### [NEW] Inconsistent naming for `ValidatorProxy.operator` variable

#### Description

`ValidatorProxy.operator` at ValidatorProxy.sol#L19 is actually a `nodeOperatorRegistry`. The same goes for a function `setOperator` at ValidatorProxy.sol#L54

#### Recommendation

We recommend renaming `operator` variable and function `setOperator` to `operatorRegistry` and `setOperatorRegistry` respectively.

### [NEW] Misleading revert message in case of pending funds

#### Description

If there are pending funds, the user may not be able to withdraw and will get a general error "Too much to withdraw" (StMATIC.sol#L184). Please consider adding to UI that he/she should wait X days as the funds have not yet been received.

#### Recommendation

We suggest handling this case in UI.

### [NEW] Magic numbers are used

#### Description

Magic numbers at StMATIC.sol#L897 decrease code readability. A reader without context won't understand what do they mean. Also, it complicates further code maintenance.

```
uint256 exchangeRatePrecision = validatorId < 8 ? 100 : 10**29;
```

#### Recommendation

We recommend using constants with descriptive names or add a comment explaining what is happening.

## Results

| Level | Amount |
| --- | --- |
| CRITICAL | |
| MAJOR | |
| WARNING | |
| INFO | |
| Total | |

## Conclusion

The fixes introduced in this PR do fix what they should. But one major issue was found that was introduced in earlier changes. There are also a number of minor comments that may help to increase the code readability.