LIDO PROTOCOL SECURITY AUDIT REPORT

May 24, 2022

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about 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 fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Lido. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals

- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the cients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Customer with an interim report.

5. Bug fixing & re-audit:

- The Customer either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Customer with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Customer with a re-audited report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Other non-essential issues and recommendations reported to/ acknowledged by the team.

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

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

LIDO protocol is a project for staking Ether to use it in the Beacon chain. Users can deposit Ether to the Lido smart contract and receive stETH tokens in return. The stETH token balance corresponds to the amount of the Beacon chain Ether that the holder could withdraw if state transitions were enabled right now in the Ethereum 2.0 network.
The Lido DAO is a Decentralized Autonomous Organization that manages the liquid staking protocol by deciding on key parameters (e.g., setting fees, assigning node operators and oracles, etc.) through the voting power of governance token (LDO) holders.
The Lido DAO is an Aragon organization. The protocol smart contracts extend the AragonApp base contract and can be managed by the DAO.

This scope contains contracts that include the following logic:

- LIP-6: In-protocol coverage application mechanism
- LIP-7: Composite oracle beacon report receiver
- LIP-8: Increase keysOpIndex in assignNextSigningKeys
- LIP-9: Add an explicit log for the stETH burn events
- LIP-10: Proxy initializations and LidoOracle upgrade
- LIP-11: Add a transfer shares function for stETH
- LIP-12: On-chain part of the rewards distribution after the Merge
- LIP-14: Protocol safeguards. Staking rate limiting
- LIP-15: Protocol safeguards. Resume role

Main interacting contracts:
`Lido` - The contract contains the basic logic for accepting deposits from Ethereum 1.0 and transferring them to the Beacon chain.
`NodeOperatorsRegistry` - This contract stores the data on deposits of all validators that will be registered by the protocol. Each deposit data record contains, among other things, the public key of the validator.

`DepositSecurityModule` - This is a contract created for committee members and the safe call of the `depositBufferedEther()` function in Lido's contract.

`deposit_contract` - This contract is required for the Beacon chain deposits and is made according to the specification eth2.0-specs. It hasn't been changed.
`OrderedCallbacksArray` - This defines an ordered callbacks array supporting add/insert/remove ops.
`CompositePostRebaseBeaconReceiver` - This defines a composite post-rebase beacon receiver for the Lido oracle.
`SelfOwnedStETHBurner` - This is a dedicated contract for enacting stETH burning requests.
`StETH` - This contract implements the logic for the StETH token. StETH balances are dynamic and represent the holder's share in the total amount of Ether controlled by the protocol.
`WstETH` - This contract implements the logic for the WstETH token. This is a StETH token wrapper with static balances. WstETH token's balance only changes on transfers, unlike StETH that is also changed when oracles report staking rewards and penalties.
`LidoExecutionLayerRewardsVault.sol` - This is a vault for temporary storage of the execution layer rewards (MEV and transaction fees).
`ECDSA` - This is a library for getting an address from the encoded data. Imported from OpenZeppelin.
`StakeLimitUtils` - The helper library to implement stake rate limiting low-level calculations.
`MemUtils`, `Pausable`, `BytesLib`, `ReportUtils` - These contracts contain various helper libraries.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Lido
Project name	Lido Protocol
Interim audit timeline (Core Protocol)	January 31, 2022 - February 17, 2022
Interim audit timeline (Stake limit)	April 25, 2022 - April 27, 2022
Interim audit timeline (Merge-ready protocol code with final changes)	May 19, 2022 - May 23, 2022

Title	Description
Number of auditors	4

Project Log

Date	Commit Hash	Note
31-01-2022	801d3e854efb33ff33a59fe51187e187047a6be2	Initial audit.
28-02-2022	b1fd2dc424567f206ea690e1a708f4584fe5231f	Reaudit based on the fixes provided.
25-04-2022	2c54f043c22cd8a23e8b8407ba7abff2c0f681ba	Additional stake limit audit.
19-05-2022	08436ce13d67501fa723169c1dc69fe47b90cde4	Merge-ready protocol code audit. The LidoMevTxFeeVault contract has been renamed to LidoExecutionLayerRewardsVault.

Project Scope

The audit covered the following files:

File name	Link
NodeOperatorsRegistry.sol	NodeOperatorsRegistry.sol
StETH.sol	StETH.sol
Lido.sol	Lido.sol
ReportUtils.sol	ReportUtils.sol
LidoOracle.sol	LidoOracle.sol
MemUtils.sol	MemUtils.sol

File name	Link
Pausable.sol	Pausable.sol
deposit_contract.sol	deposit_contract.sol
WstETH.sol	WstETH.sol
DepositSecurityModule.sol	DepositSecurityModule.sol
ECDSA.sol	ECDSA.sol
CompositePostRebaseBeaconReceiver.sol	CompositePostRebaseBeaconReceiver.sol
OrderedCallbacksArray.sol	OrderedCallbacksArray.sol
SelfOwnedStETHBurner.sol	SelfOwnedStETHBurner.sol
LidoExecutionLayerRewardsVault.sol	LidoExecutionLayerRewardsVault.sol
StakeLimitUtils.sol	StakeLimitUtils.sol

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	1
Medium	7
Low	7

ID	Name	Severity	Status
H-1	Opportunity to add bufferedETH without submitting to LIDO	High	Fixed
M-1	Extra function	Medium	Fixed
M-2	No validation of the address parameter value in the contract constructor	Medium	Fixed
M-3	Max oracle members amount is actually lower	Medium	Fixed
M-4	There is no recovery option for ERC721	Medium	Fixed
M-5	Incorrect event	Medium	Fixed
M-6	There is no recovery for excess `stETH`	Medium	Acknowledged
M-7	Callback verification	Medium	Fixed
L-1	No check before initialization	Low	Fixed
L-2	No comparison with previous value	Low	Fixed
L-3	A comment about the node operator count	Low	Fixed
L-4	Not all params are there at the comment	Low	Fixed
L-5	Code inconsistency	Low	Fixed
L-6	Typo mistake	Low	Fixed
L-7	Using `transferShares()` is possible	Low	Acknowledged

1.6 Conclusion

Smart contracts have been audited and several suspicious places have been detected. During the audit, no critical vulnerabilities were found. One high, seven medium, and seven low issues were identified. After working on the reported findings, all of them were confirmed and fixed by the client and two findings were acknowledged.

Final commit identifier with all fixes: `08436ce13d67501fa723169c1dc69fe47b90cde4`

File name	Contract deployed on mainnet
Lido.sol	0x47EbaB13B806773ec2A2d16873e2dF770D130b50
WstETH.sol	0x7f39c581f595b53c5cb19bd0b3f8da6c935e2ca0
LidoOracle.sol	0x1430194905301504e8830ce4B0b0df7187E84AbD
NodeOperatorsRegistry.sol	0x5d39ABaa161e622B99D45616afC8B837E9F19a25
LidoExecutionLayerRewardsVault.sol	0x388C818CA8B9251b393131C08a736A67ccB19297
DepositSecurityModule.sol	0x710B3303fB508a84F10793c1106e32bE873C24cd
CompositePostRebaseBeaconReceiver.sol	0x55a7E1cbD678d9EbD50c7d69Dc75203B0dBdD431
SelfOwnedStETHBurner.sol	0xB280E33812c0B09353180e92e27b8AD399B07f26
deposit_contract.sol	0x00000000219ab540356cbb839cbe05303d7705fa

2. FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

H-1	Opportunity to add bufferedETH without submitting to LIDO
File	LidoMevTxFeeVault.sol#L79
Severity	High
Status	Fixed at e3b84476

Description

It is possible to send ETH to the `LidoMevTxFeeVault` contract and when the oracle reports contract sends ETH to LIDO, which will be used to rewards, it may fluctuate the price of lido shares.
LidoMevTxFeeVault.sol#L79

Recommendation

We recommend that the `LidoMevTxFreeVault` contract should receive ETH only from authorized addresses or the `withdrawRewards()` function should have limits.

2.3 Medium

M-1	Extra function
File	Lido.sol#L139-L142
Severity	Medium
Status	Fixed at 2f49e4b1

Description

At lines Lido.sol#L139-L142 and at lines Lido.sol#L158-L165 similar actions are performed.
But in one case there is a change in the `TOTAL_MEV_TX_FEE_COLLECTED_POSITION` variable, and in the other case there is none.
One of these functions is redundant.

Recommendation

Need to remove the redundant feature.

M-2	No validation of the address parameter value in the contract constructor
File	LidoMevTxFeeVault.sol#L72
Severity	Medium
Status	Fixed at c8b065c1

Description

The variable is assigned the value of the constructor input parameter. But this parameter is not checked before this. If the value turns out to be zero, then it will be necessary to redeploy the contract, since there is no other functionality to set this variable.
At line LidoMevTxFeeVault.sol#L72 the `TREASURY` variable is set to the value of the `_treasury` input parameter.

Recommendation

It is necessary to add a check of the input parameter to zero before initializing the variable.

M-3	Max oracle members amount is actually lower
File	LidoOracle.sol#L432
Severity	Medium
Status	Fixed at 2c592fe8

Description

Max members are 255 instead of 256 which may affect the quorum: LidoOracle.sol#L432

Recommendation

The check needs to be corrected.

M-4	There is no recovery option for ERC721
File	Lido.sol#L356
Severity	Medium
Status	Fixed at ea5e2322

Description

The `transferToVault()` function doesn't support ERC721 tokens: Lido.sol#L356

Recommendation

It is necessary to add another function to recover ERC721.

M-5	Incorrect event
File	StETH.sol#L461
Severity	Medium
Status	Fixed at f8b4b96e

Description

In case of cover, `stETH` doesn't burn StETH.sol#L461

Recommendation

It is necessary to exclude the `stETH` amount from the event.

M-6	There is no recovery for excess `stETH`
File	WstETH.sol#L28-L118
Severity	Medium
Status	Acknowledged

Description

It is possible to transfer `stETH` to `wstETH` so it will be frozen in the contract. WstETH.sol#L28-L118

Recommendation

It is necessary to add a function to recover excess `stETH` and keep the wrapped shares amount.

Client's commentary

It's hard to add the recovery function easily to the wstETH contract because it isn't upgradable.
In case of emergency it's possible to update the Lido contract with a specific workaround for wstETH recovering stETH from the wstETH contract.

M-7	Callback verification
Files	OrderedCallbacksArray.sol#L60 LidoOracle.sol#L644
Severity	Medium
Status	Fixed at 9babc852

Description

By mistake a callback which has no implementation of the`processLidoOracleReport()` method can be added at the line: OrderedCallbacksArray.sol#L60 in case you set the `IBeaconReportReceiver` address, the execution of the following lines will be reverted. LidoOracle.sol#L644

Recommendation

It is necessary to add verification of the existing `processLidoOracleReport()` method in callback or callbacks should be double-checked before adding.
See this standard: https://eips.ethereum.org/EIPS/eip-165.

2.4 Low

L-1	No check before initialization
File	Lido.sol#L339-L340
Severity	Low
Status	Fixed at 9985ffbf

Description

At the lines Lido.sol#L339-L340 If the value of variable `mevRewards` is equal to 0, then the initialization of variable `BUFFERED_ETHER_POSITION` will still be performed. This will also require gas consumption.

Recommendation

It is recommended to add a check before initialization.

L-2	No comparison with previous value
File	DepositSecurityModule.sol#L204
Severity	Low
Status	Fixed at fdd6b7a7

Description

At line DepositSecurityModule.sol#L204 the variable is initialized. But if the new value is equal to the old value, the excess gas will be wasted.

Recommendation

It is recommended to add a check before initializing the variable.

L-3	A comment about the node operator count
File	NodeOperatorsRegistry.sol#L23
Severity	Low
Status	Fixed at fb8bafe3

Description

There is a comment about the node operator count but there is no functionality related to it at the line: NodeOperatorsRegistry.sol#L23
Only manual moderating is available.

Recommendation

It is recommended to add a check for the maximum number of operators when adding new ones.

L-4	Not all params are there at the comment
File	Lido.sol#L97
Severity	Low
Status	Fixed at 6adfa826

Description

After line Lido.sol#L97 a description of the two parameters `_treasury` and `_insuranceFund` must be added.

Recommendation

It is recommended to fix the code.

L-5	Code inconsistency
File	Lido.sol#L111
Severity	Low
Status	Fixed at c0ad70b3

Description

The address above is not cast to address Lido.sol#L111.

Recommendation

It is recommended to fix the code.

L-6	Typo mistake
File	LidoOracle.sol#L78
Severity	Low
Status	Fixed at 66850e2f

Description

It should be `stored` LidoOracle.sol#L78

Recommendation

It is recommended to fix the code.

L-7	Using `transferShares()` is possible
File	WstETH.sol#L73
Severity	Low
Status	Acknowledged

Description

Transfer can be made via `transferShares()`: WstETH.sol#L73

Recommendation

It is recommended to fix the code.

Client's commentary

Acknowledged.
We can't upgrade wstETH in a straightforward way, so the cost of using the old function is less than the cost of re-integrating the new implementation from scratch.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and
software testing solutions, do research and tech consultancy.

Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes