LIDO ORACLE SMART CONTRACT AUDIT

April 26, 2021

MixBytes()

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 PROJECT OVERVIEW

LIDO protocol is a project for stacking Ether to use it in 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 Beacon chain Ether that the holder could withdraw if state transitions were enabled right now in the Ethereum 2.0 network. In second version of Lido Oracle's smart contract developers add logical consistency check of the Lido changes in Ether 2.0 balance in Beacon chain

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

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:

Status	Description
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.

1.4 EXECUTIVE SUMMARY

LidoOracle is a contract where oracles send addresses' balances controlled by the DAO on the ETH 2.0 side. The balances can go up because of reward accumulation and can go down due to slashing and staking penalties.

1.5 PROJECT DASHBOARD

Client	Lido
Audit name	Oracle
Initial version	34166528649d6873ad13a939d93cc212907bfc83
Final version	bea1be988a6f5db1dc2bd002ab5919b71e4ce796
SLOC	435
Date	2021-04-12 - 2021-04-26
Auditors engaged	2 auditors

FILES LISTING

LidoOracle.sol	LidoOracle.sol
ReportUtils.sol	ReportUtils.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	4
Comment	3

CONCLUSION

Smart contract has been audited and several suspicious places were found. During audit no critical and major issues were identified. Several issues were marked as warning and comments. After working on audit report all issues were fixed or acknowledged(if issue is not critical or major) by client, so contracts assumed as secure to use according our security criteria. Final commit identifier with all fixes: `bea1be988a6f5db1dc2bd002ab5919b71e4ce796`

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Incorrect mass indexation
File	LidoOracle.sol
Severity	Warning
Status	**No issue**

DESCRIPTION

If user or another smart contract invoke the following function with incorrect index than tx fail: LidoOracle.sol#L212

RECOMMENDATION

We recommend to add simple check to give some information if revert occures:

```
require(_index < currentReportVariants.length, "INCORRECT_INDEX");
```

CLIENT'S COMMENTARY

Solidity already provides array bounds checking so there's no risk of incorrect behavior. We could add an explicit check but that would consume an extra gas. The contract source is validated on Etherscan so one could use freely available tools like Tenderly to check the exact place in code that triggered the revert.

WRN-2	Possible wrong initialization
File	LidoOracle.sol
Severity	Warning
Status	**No issue**

DESCRIPTION

Initialization of smart contract has public access which can be used to wrong initialization of `ALLOWED_BEACON_BALANCE_ANNUAL_RELATIVE_INCREASE_POSITION` and `ALLOWED_BEACON_BALANCE_RELATIVE_DECREASE_POSITION`: LidoOracle.sol#L341

RECOMMENDATION

We recommend to add some role which can invoke `initialize_v2` function.

CLIENT'S COMMENTARY

The `initialize_v2` function is called atomically as a part of the upgrade transaction. The call is allowed only once, so there's no way to call it with unintended arguments afterwards.

WRN-3	Possible division by zero
File	LidoOracle.sol
Severity	Warning
Status	**No issue**

DESCRIPTION

If some storage variables are not initialized then in the following functions occurs revert due to division by zero: LidoOracle.sol#L656, LidoOracle.sol#L663

RECOMMENDATION

We recommend to add a simple check:

```
require(_beaconSpec.epochsPerFrame > 0, "SPEC_NOT_INITIALIZED");
```

CLIENT'S COMMENTARY

The Beacon spec either won't change or will change dramatically, so if the change is needed it won't be performed mindlessly. The required validation will be much more complex than checking for zeroes and will be done during the likely upgrade. Apart from that, division by zero will revert the transaction and won't break the protocol correctness.

WRN-4	Possible front-tunning attack
File	LidoOracle.sol
Severity	Warning
Status	**No issue**

DESCRIPTION

In current version of smart contract if `MANAGE_QUORUM` conspire with one of the oracles and set `QUORUM_POSITION = 1` then they can change Beacon balance for maximum allowed value for next epochs: LidoOracle.sol#L406

RECOMMENDATION

We recommend using a very strict policy for `MANAGE_QUORUM`.

CLIENT'S COMMENTARY

The `MANAGE_QUORUM` permission is only assigned to the DAO Voting contract. Thus, quorum updates can only be made by LDO holders collectively via DAO voting.

2.4 COMMENTS

CMT-1	Possible overflow
File	LidoOracle.sol
Severity	Comment
Status	No issue

DESCRIPTION

Incorrect Beacon Spec's can lead to overflow in the following function:
LidoOracle.sol#L301

RECOMMENDATION

We recommend to use safemath for mathematical operations.

CLIENT'S COMMENTARY

The Beacon spec configuration values cannot be changed in an unauthorized way, so the check that the multiplication won't overflow will be performed as part of the upgrade, if one is needed.

CMT-2	Insufficient information in revert
File	LidoOracle.sol
Severity	Comment
Status	**No issue**

DESCRIPTION

If the old Beacon Spec is used, then `reportBeacon` function should contain information about it here: LidoOracle.sol#L433

RECOMMENDATION

We recommend to add more information to `require` message.

CLIENT'S COMMENTARY

The concern was that the Beacon spec may change on Beacon chain side, in which case a more informational revert reason would be desired. This is not currently possible since we cannot distinguish between the two cases: 1) the Beacon spec has changed on Beacon chain side (which, we believe, is very unlikely to happen) and 2) the oracle has reported an incorrect epoch.

CMT-3	Implicit math
File	LidoOracle.sol
Severity	Comment
Status	**Fixed** at **bea1be98**

DESCRIPTION

There is implicit increment for `currentReportVariants` (LidoOracle.sol#L459):

```
++currentReportVariants[i];
```

This is sum of first bits:

```
 * +00 | uint16 | count
 * +16 | uint32 | beaconValidators
 * +48 | uint64 | beaconBalance
```

RECOMMENDATION

We recommend to add comment about math.

3.ABOUT MIXBYTES

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

BLOCKCHAINS

Ethereum
Cosmos
EOS
Substrate

TECH STACK

Python
Solidity
Rust
C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes