LIDO ORACLE SECURITY AUDIT REPORT

Oct 31, 2024

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION 2

1.1 Disclaimer 2

1.2 Security Assessment Methodology 2

1.3 Project Overview 6

1.4 Project Dashboard 7

1.5 Summary of findings 16

1.6 Conclusion 17

2.FINDINGS REPORT 18

2.1 Critical 18

2.2 High 18

2.3 Medium 18

2.4 Low 18

L-1 Potential Revert When `available_eth == 0` 18

L-2 Incorrect comment for `FrameCheckpoint` 20

L-3 Incorrect Calculation of `finalized_epoch` in `CSOracle.collect_data()` 21

3. ABOUT MIXBYTES 22

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 the Client. 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 clients' 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 Client with an interim report.

5. Bug fixing & re-audit:

- The Client 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 Client 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

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit 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 of funds.
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	Bugs that do not have a significant immediate impact and could be easily fixed.

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

The Lido Oracle is a daemon for the Lido decentralized staking service, consisting of three primary modules:

1. Accounting Module

    - Updates protocol TVL and distributes node-operator rewards
    - Manages exited/stuck validator information and user withdrawal requests
    - Controls bunker mode activation
    - Operates in 24-hour frames with stages: waiting, data collection, hash consensus, core update report, and extra data report

2. Ejector Module

    - Manages Lido validator exits based on withdrawal needs
    - Predicts ETH income and schedules validator withdrawals
    - Can force eject validators from Node Operators with boosted exits

3. CSM (Consensus Staking Module) Module

    - Collects CSM node operator performance data
    - Calculates validator performance based on attestations
    - Creates a Merkle tree of operator share distribution
    - Submits the Merkle root to the module contract

These modules maintain the Lido staking service's integrity, ensuring accurate reward distribution, validator management, and performance tracking.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Lido
Project name	Lido Oracle
Timeline	23.09.2024 - 31.10.2024
Number of Auditors	3

Project Log

Date	Commit Hash	Note
23.09.2024	4e1e2210483fb44926d751049ea2d21561779dc8	Commit for the audit (Lido Oracle)
23.09.2024	f4ad6e006b8daf05ce2ce255e123eb9f923d8ef8	Commit for the audit (oz-merkle-tree)
08.10.2024	f3b33f8e8281eb9f08ed396c8c84b4936f2be888	Commit for the re-audit
10.10.2024	93831435c5ca787f4f92dc9416788e0cdf5e3f4b	Commit with updates
17.10.2024	3cc1193df61068f32504c2913f1f3da8bd179362	Commit with updates
30.10.2024	2a0ecd103509814c99326d9a7a5bf9099d708504	Commit with updates

Project Scope

The audit covered the following files:

File name	Link
src/__init__.py	__init__.py
src/constants.py	constants.py
src/main.py	main.py
src/metrics/__init__.py	__init__.py
src/metrics/healthcheck_server.py	healthcheck_server.py
src/metrics/logging.py	logging.py
src/metrics/prometheus/__init__.py	__init__.py
src/metrics/prometheus/accounting.py	accounting.py
src/metrics/prometheus/basic.py	basic.py
src/metrics/prometheus/business.py	business.py
src/metrics/prometheus/csm.py	csm.py
src/metrics/prometheus/duration_meter.py	duration_meter.py
src/metrics/prometheus/ejector.py	ejector.py
src/metrics/prometheus/validators.py	validators.py
src/modules/__init__.py	__init__.py
src/modules/accounting/__init__.py	__init__.py
src/modules/accounting/accounting.py	accounting.py
src/modules/accounting/third_phase/__init__.py	__init__.py
src/modules/accounting/third_phase/extra_data.py	extra_data.py

File name	Link
src/modules/accounting/third_phase/extra_data_v2.py	extra_data_v2.py
src/modules/accounting/third_phase/types.py	types.py
src/modules/accounting/types.py	types.py
src/modules/checks/__init__.py	__init__.py
src/modules/checks/checks_module.py	checks_module.py
src/modules/checks/pytest.ini	pytest.ini
src/modules/checks/suites/__init__.py	__init__.py
src/modules/checks/suites/common.py	common.py
src/modules/checks/suites/conftest.py	conftest.py
src/modules/checks/suites/consensus_node.py	consensus_node.py
src/modules/checks/suites/execution_node.py	execution_node.py
src/modules/checks/suites/ipfs.py	ipfs.py
src/modules/checks/suites/keys_api.py	keys_api.py
src/modules/csm/__init__.py	__init__.py
src/modules/csm/checkpoint.py	checkpoint.py
src/modules/csm/csm.py	csm.py
src/modules/csm/log.py	log.py
src/modules/csm/state.py	state.py

File name	Link
src/modules/csm/tree.py	tree.py
src/modules/csm/types.py	types.py
src/modules/ejector/__init__.py	__init__.py
src/modules/ejector/data_encode.py	data_encode.py
src/modules/ejector/ejector.py	ejector.py
src/modules/ejector/types.py	types.py
src/modules/submodules/__init__.py	__init__.py
src/modules/submodules/consensus.py	consensus.py
src/modules/submodules/exceptions.py	exceptions.py
src/modules/submodules/oracle_module.py	oracle_module.py
src/modules/submodules/types.py	types.py
src/providers/__init__.py	__init__.py
src/providers/consensus/__init__.py	__init__.py
src/providers/consensus/client.py	client.py
src/providers/consensus/types.py	types.py
src/providers/consistency.py	consistency.py
src/providers/execution/__init__.py	__init__.py
src/providers/execution/base_interface.py	base_interface.py
src/providers/execution/contracts/accounting_oracle.py	accounting_oracle.py

File name	Link
src/providers/execution/contracts/base_oracle.py	base_oracle.py
src/providers/execution/contracts/burner.py	burner.py
src/providers/execution/contracts/cs_accounting.py	cs_accounting.py
src/providers/execution/contracts/cs_fee_distributor.py	cs_fee_distributor.py
src/providers/execution/contracts/cs_fee_oracle.py	cs_fee_oracle.py
src/providers/execution/contracts/cs_module.py	cs_module.py
src/providers/execution/contracts/deposit_contract.py	deposit_contract.py
src/providers/execution/contracts/exit_bus_oracle.py	exit_bus_oracle.py
src/providers/execution/contracts/hash_consensus.py	hash_consensus.py
src/providers/execution/contracts/lido.py	lido.py
src/providers/execution/contracts/lido_locator.py	lido_locator.py
src/providers/execution/contracts/oracle_daemon_config.py	oracle_daemon_config.py
src/providers/execution/contracts/oracle_report_sanity_checker.py	oracle_report_sanity_checker.py
src/providers/execution/contracts/staking_router.py	staking_router.py
src/providers/execution/contracts/withdrawal_queue_nft.py	withdrawal_queue_nft.py

File name	Link
src/providers/execution/exceptions.py	exceptions.py
src/providers/http_provider.py	http_provider.py
src/providers/ipfs/__init__.py	__init__.py
src/providers/ipfs/cid.py	cid.py
src/providers/ipfs/dummy.py	dummy.py
src/providers/ipfs/gw3.py	gw3.py
src/providers/ipfs/multi.py	multi.py
src/providers/ipfs/pinata.py	pinata.py
src/providers/ipfs/public.py	public.py
src/providers/ipfs/types.py	types.py
src/providers/keys/__init__.py	__init__.py
src/providers/keys/client.py	client.py
src/providers/keys/types.py	types.py
src/services/__init__.py	__init__.py
src/services/bunker.py	bunker.py
src/services/bunker_cases/__init__.py	__init__.py
src/services/bunker_cases/abnormal_cl_rebase.py	abnormal_cl_rebase.py
src/services/bunker_cases/midterm_slashing_penalty.py	midterm_slashing_penalty.py
src/services/bunker_cases/types.py	types.py

File name	Link
src/services/exit_order/__init__.py	__init__.py
src/services/exit_order/iterator.py	iterator.py
src/services/exit_order/iterator_state.py	iterator_state.py
src/services/exit_order_v2/__init__.py	__init__.py
src/services/exit_order_v2/iterator.py	iterator.py
src/services/prediction.py	prediction.py
src/services/safe_border.py	safe_border.py
src/services/validator_state.py	validator_state.py
src/services/withdrawal.py	withdrawal.py
src/types.py	types.py
src/utils/__init__.py	__init__.py
src/utils/abi.py	abi.py
src/utils/blockstamp.py	blockstamp.py
src/utils/build.py	build.py
src/utils/cache.py	cache.py
src/utils/dataclass.py	dataclass.py
src/utils/env.py	env.py
src/utils/events.py	events.py
src/utils/exception.py	exception.py
src/utils/input.py	input.py

File name	Link
src/utils/range.py	range.py
src/utils/slot.py	slot.py
src/utils/timeit.py	timeit.py
src/utils/types.py	types.py
src/utils/validator_state.py	validator_state.py
src/utils/web3converter.py	web3converter.py
src/variables.py	variables.py
src/web3py/__init__.py	__init__.py
src/web3py/contract_tweak.py	contract_tweak.py
src/web3py/extensions/__init__.py	__init__.py
src/web3py/extensions/consensus.py	consensus.py
src/web3py/extensions/contracts.py	contracts.py
src/web3py/extensions/csm.py	csm.py
src/web3py/extensions/fallback.py	fallback.py
src/web3py/extensions/keys_api.py	keys_api.py
src/web3py/extensions/lido_validators.py	lido_validators.py
src/web3py/extensions/tx_utils.py	tx_utils.py
src/web3py/middleware.py	middleware.py
src/web3py/types.py	types.py
src/__init__.py	__init__.py

File name	Link
src/oz_merkle_tree/__init__.py	__init__.py
src/oz_merkle_tree/base.py	base.py
src/oz_merkle_tree/tree.py	tree.py

Deployments

After the audit conducted, we verified that the Dockerfile used to build an image works correctly and uses the audited sorce code. Also, the published image with the digest `sha256:5ab6398e68440c009dca372bac4de9a30c18b9afaa64e32d7a3ff4f7c6f0858c` corresponds to the audited scope.

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	0
Medium	0
Low	3

ID	Name	Severity	Status
L-1	Potential Revert When `available_eth == 0`	Low	Fixed
L-2	Incorrect comment for `FrameCheckpoint`	Low	Fixed
L-3	Incorrect Calculation of `finalized_epoch` in `CSOracle.collect_data()`	Low	Acknowledged

1.6 Conclusion

During the audit, we thoroughly tested critical attack vectors and verified the following:

- All instances of the off-chain service will be collecting report data for the same block, therefore they should send the exact same data;
- Extra data submission will correctly work in case the predefined oracle service will unexpectedly revert during submission. This is enforced by the usage of the `check_and_send_transaction` that will allow other instances to send the data;
- Contract version update is correctly handled by the off-chain service;
- `get_all_node_operator_digests` will correctly work even if CSM module has a lot of node operators by batching the request;
- `COMPATIBLE_ONCHAIN_VERSIONS` list in each module correctly accounts for the contract and consensus version set in the contracts;
- The amount of ejected validators is correctly calculated: the currently available ETH balance together with the predicted ETH balance (which utilizes `prediction_duration_in_slots` parameter from the `OracleDaemonConfig`) are calculated to determine the minimum number of validators that will be sufficient to complete unfinalized withdrawal requests;
- Validators to be ejected are correctly selected from the list of all available Lido validators. `ValidatorExitIteratorV2` uses sufficient predicates to sort validators in ascending order;
- StETH sent directly to the `CSFeeDistributor` contract is correctly accounted for by the CSM oracle;
- When multiple frames are skipped, the CSM oracle correctly calculates rewards for the entire missed period;

2. FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

Not Found

2.4 Low

L-1	Potential Revert When `available_eth == 0`
Severity	Low
Status	Fixed in f3b33f8e

Description

The issue is identified in the `_calculate_finalization_batches` method of the `Withdrawal` service. If `available_eth == 0`, the call to `calculate_finalization_batches` will revert due to internal checks. This can cause the entire function to fail, even though it could handle the zero available ETH case more gracefully.

The issue is classified as **Low** severity because it occurs in a specific edge case, where no ETH is available for finalization. However, addressing this would improve the robustness of the service, especially in low-ETH conditions.

Recommendation

We recommend adding a check for `available_eth == 0` before calling the `calculate_finalization_batches` function. If `available_eth` is zero, the function should return an empty list or handle this scenario in a way that prevents reverts.

L-2	Incorrect comment for `FrameCheckpoint`
Severity	Low
Status	Fixed in f3b33f8e

Description

`FrameCheckpoint.slot` is the first slot of `max(duty_epochs) + 2` epoch, not checkpoint.py#L29.

Recommendation

We recommend clarifying the comment.

L-3	Incorrect Calculation of `finalized_epoch` in `CSOracle.collect_data()`
Severity	Low
Status	Acknowledged

Description

When calculating the last finalized epoch, csm.py#L180 that `blockstamp.slot_number` is the first slot of the justifying epoch. However, if the first slot of the justifying epoch is empty, `blockstamp.slot_number` oracle_module.py#L81-L88 to the slot where the last finalized block was created. As a result, `finalized_epoch` in this case will be less than the actual number of the last finalized epoch.

Recommendation

We recommend considering the case when the first slot of the justifying epoch is empty.

Client's commentary

> The proper handling of this edge case would require a more complex logic implementation. In this case, the current simplified algorithm might process the report with a delay of one epoch at worst. The simpler solution is easier to implement and easier to maintain.

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